CATEGORIE TECHNIQUE
Rue Peetermans, 80 – 4100 Seraing

Réalisation d'un logiciel d'aide à la conduite de véhicule sur banc à rouleaux (Utilisation de cycles normalisés)

Jimmy CLAIRBOIS

Travail de fin d'études présenté en vue de l'obtention du titre de Bachelier en informatique et systèmes, finalité informatique industrielle

ANNEE ACADEMIQUE
2008-2009





Remerciements

Madame Cécile Moitroux, (ingénieur Civil Electricien [Informatique], Maître-assistant de la HEPL) pour m'avoir guidé dans les grandes lignes du développement, ses conseils ainsi que pour la correction de ce travail de fin d'études.

Monsieur Marc Nélis (ingénieur de développement), mon maître de stage, pour son soutien, la coordination de mon projet au sein de son équipe, les choix stratégiques et ses précieux conseils.

Monsieur Guillaume Doyen (technicien préparateur – opérateur du banc moteur) pour la réalisation des tests en conditions réelles de mon logiciel sur le banc à rouleaux, de ses conseils et aides quand à la compréhension du fonctionnement du banc.

Monsieur Pierre Detré (chef de projet ECOMobile) pour son aide sur la compréhension du sujet ainsi que son aide quand à la réalisation du simulateur électrique de débitmètre.

Monsieur Jonathan Courtens (project engineer) pour son aide concernant le débitmètre.

À l'équipe ECOMobile dans son ensemble pour leur excellente sympathie et leur professionnalisme.

À mon binôme Fabian Crémers pour son aide durant l'année scolaire et durant mon stage.

Aux étudiants de ma promotion pour nos échanges de conseils techniques, astuces et problèmes concernant nos stages.

Aux étudiants effectuant leur TFE au sein du campus et, plus particulièrement à Cédric Gilles, collègue de travail, pour tous les bons moments.

À la neige bloquant l'autoroute et me laissant embourbé sur le parking du campus.

Aux nombreuses courses sur le circuit qui ont égayé mes journées.

Table des matières





1 Introduction

Le stage s'effectue principalement au Campus Automobile de Spa-Francorchamps qui est situé en bordure intérieure du circuit automobile.

C'est un centre de compétences possédant le statut d'Association Sans But Lucratif qui a vu le jour en 2006.

Etant passionné du monde de la compétition moteur, je visite régulièrement les sites internet touchant à ce domaine. C'est en visitant le site internet du campus que j'ai vu les diverses propositions de travaux de fin d'étude et notamment celui que j'ai choisi, l'unique TFE dans le domaine informatique.

Malheureusement, une des conditions pour réaliser un stage en entreprise est que ladite entreprise doit employer un informaticien qui soit disponible pour aider l'étudiant en cas de problème. Or, le campus ne possède pas d'informaticien.

De ce fait, j'ai brièvement hésité en vue des nombreuses inconnues techniques pour lesquelles j'aurai dû trouver la solution par moi-même, sans aucune aide.

Finalement, aimant plus que tout relever des défis et avec l'accord de Mme Moitroux, j'ai décidé de choisir ce sujet pour mon travail de fin d'études.

Le sujet consiste en la réalisation d'un logiciel d'aide à la conduite de véhicule sur banc à rouleaux en vue de suivre un cycle de conduite normalisé.

Pour ce faire, il faut :

- Acquérir des données depuis le banc à rouleaux.
- Afficher les informations en temps réel. (à un certain intervalle)
- Générer le rapport concernant le cycle.

2 Présentation de l'association

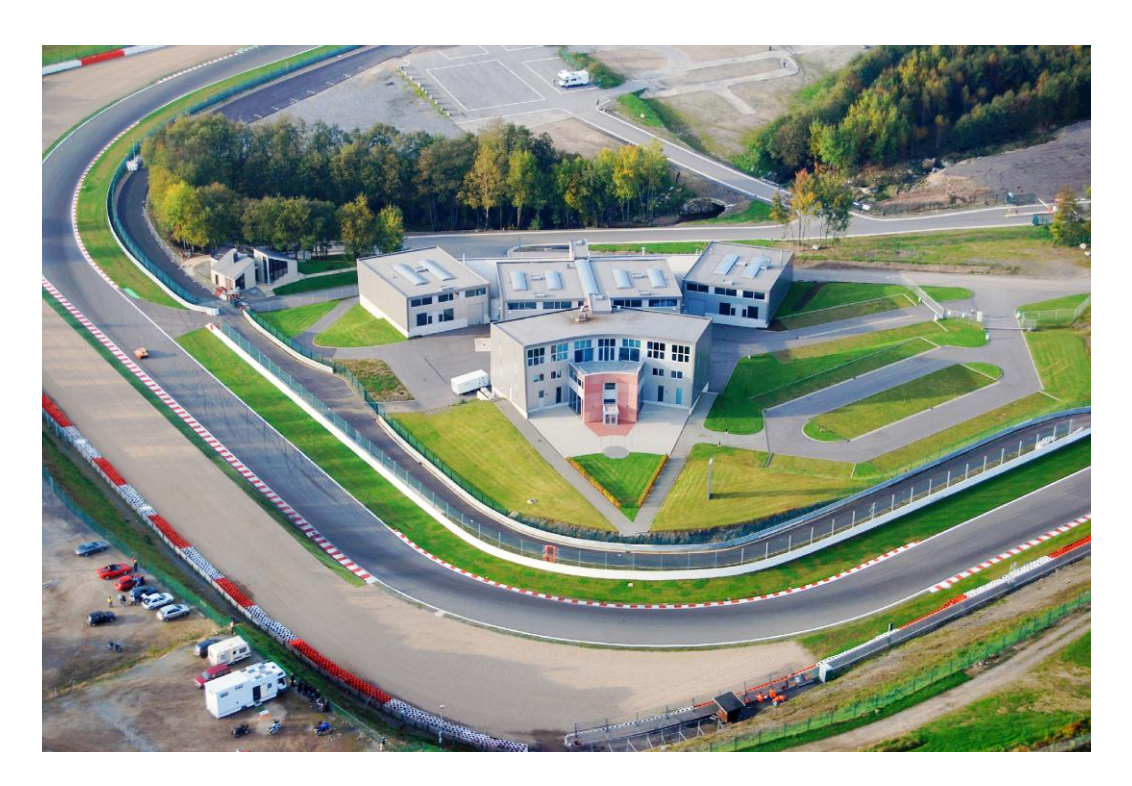


Figure 2-1 Campus Automobile – Vue aérienne

2.1 Généralités

Le Campus Automobile de Spa-Francorchamps est un centre de compétence spécialisé dans le domaine automobile qui se veut être une référence et une vitrine pour le monde automobile actuel et futur.

Son rôle est essentiellement de proposer un panel de formations pouvant déboucher dans le métier automobile mais aussi la recherche dans les nouvelles technologies, les sensibilisations ainsi que la fourniture de services aux entreprises.

Construit en bordure du circuit, le Campus Automobile possède un accès direct à la piste afin de faire des tests en conditions réelles. Celle-ci est généralement libre pour le campus en soirée.

2.1.1 Détails

- ***Nom :*** Campus Automobile de Spa-Francorchamps
- ***Date de création :*** 19 janvier 2005 (tribunal de commerce de Verviers)
- ***Lieu :***
  Route du Circuit, 70
  4970 Francorchamps
- ***Type :*** association sans but lucratif
- ***Directeur :*** Paul SIMAR
- ***Web :*** www.formation-campus-automobile.be
- ***Mail :*** campus-automobile@forem.be
- ***Tél :*** 087 47 90 60
- ***Fax :*** 087 47 90 61

2.1.2 Financement

- La région wallonne.
- L'Europe dans le cadre du projet Interreg.
- Le Fond social européen.

2.1.3 Partenaires

- Université de Liège et Université d'Aachen – IKA.
- Les industries technologiques – Agoria.
- La Société de Promotion du Circuit de Francorchamps.
- Le Forem.



Figure 2-2 Campus Automobile - Vue aérienne

2.2 Domaines d'activités

2.2.1 La formation aux technologies de l'automobile

En tant que centre de compétence, le Campus Automobile offre des formations variées telles que:

- technicien de compétition en autos/motos.
- technologies de pointe.
- design.
- usinage de moteurs.
- soudage.
- matériaux composites.
- électronique.
- acquisition de données.
- gestion moteurs.
- conception par ordinateur (DAO, CAO).
- Opérateurs et techniciens en Matériaux Composites.
- Aide-Mécaniciens.
- Soudage.
- Magasinier - Caristes – VCA.

Durée: de quelques heures à plusieurs mois.
Type: formation de type technique avec essais en condition réelles.



Figure 2-3 Ateliers - Etudiant en formation

2.2.2 ECOMobile

Inauguré le vendredi 20 février 2009, le pôle ECOMobile est le département de recherche du Campus Automobile dans le développement durable et les motorisations propres.

Ses projets sont variés mais ont tous comme but commun l'écologie et l'économie d'énergie.

C'est notamment dans ce cadre que mon travail de fin d’études est réalisé, ainsi que beaucoup d'autres tels que la Lotus Elise propulsée au gaz avec un système de récupération d'énergie cinétique.



Figure 2-4 Pôle ECOMobile - Voiture roulant au bioéthanol



Figure 2-5 Pôle ECOMobile - Inauguration

2.2.3 Le développement des entreprises

Le Campus Automobile est inscrit dans la création d'un pôle automobile à Francorchamps et, plus globalement, dans un cluster automobile au niveau wallon.

Ce pôle contiendra notamment un incubateur d'entreprise situé en bordure du Campus Automobile, ainsi que un zoning industriel. Ces éléments seront dédiés pour les entreprises ayant comme activité le secteur automobile.

Un des atouts pour les entreprises du site est le fait de pouvoir bénéficier de tout les avantages du Campus Automobile tels que ses :

- formations.
- équipements.
- infrastructures.
- relations.
- compétences.



Figure 2-6 Circuit automobile - Vue aérienne

1 - Campus Automobile et Cluster Automobiliste
2 - Futur incubateur d’entreprises
3 - Race Track Connection (pit-lane)
4 - Industrial Area (futur zoning industriel)

2.2.4 La promotion des métiers exploités au campus

2.2.4.1 Formation du corps enseignant

Destiné aux enseignants spécialisés dans le domaine automobile et souhaitant parfaire leurs connaissances ou acquérir des informations sur de nouveaux domaines.

2.2.4.2 Formation des étudiants du niveau secondaire et bachelier

Un ensemble de 15 modules de formations sont prévus pour se former sur divers aspects variés tels que l'électronique, l'électricité, les bancs d'essais, et bien d'autres.

2.2.4.3 Formule Techni-Trips

Package de découverte d'une durée de 5 jours, il couple un ensemble de formations au sein du campus, du sport ainsi que des visites culturelles liées au domaine automobile.

2.2.4.4 Animations et sensibilisation aux métiers scientifiques et techniques

Grâce aux ressources disponibles sur le campus et dans la région, un panel de sensibilisation à pu être développé dans divers domaines tels que les motorisations propres, les bancs moteurs, … .

2.2.4.5 Travaux de fin d'études

Afin de parfaire leurs équipements, le campus propose chaque année un ensemble de travaux de fin d'études qui sont extrêmement variés.

Divers exemples :

- Elaboration de banc didactique pour moteur électrique ou thermique.
- Elaboration d'une Lotus Elise équipée d'un système de récupération d'énergie cinétique.
- Elaboration d'un logiciel d'aide à la conduite en vue de collecter le niveau de pollution d'une voiture.
- Elaboration de banc à rouleaux pour moto, kart, quad, scooter.

2.3 Ateliers

2.3.1 Atelier de préparation véhicules



Figure 2-7 Ateliers - Préparation

2.3.2 Atelier Electricité-Electronique

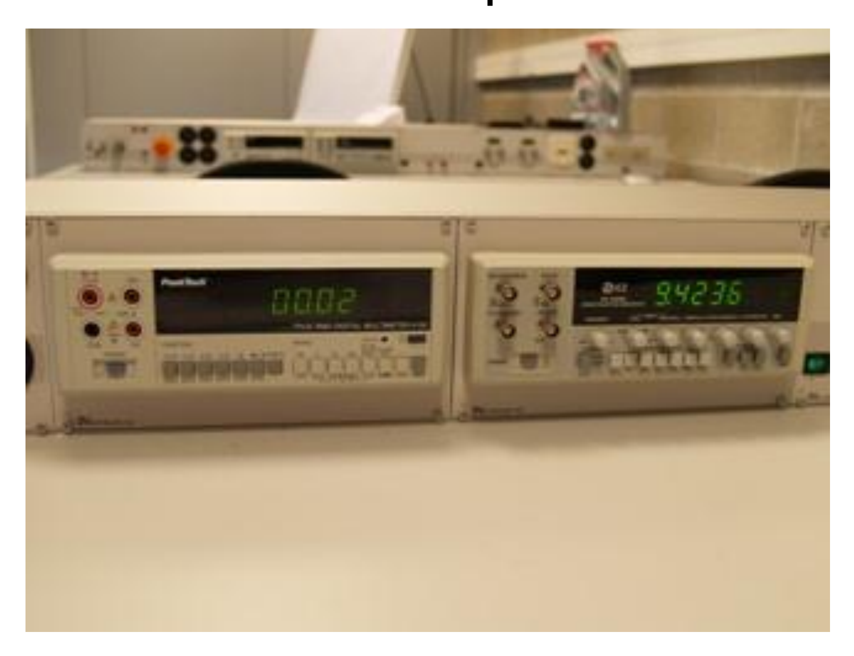


Figure 2-8 Ateliers - Electricité

2.3.3 Atelier de Métrologie



Figure 2-9 Ateliers - Métrologie

2.3.4 Atelier de soudage



Figure 2-10 Ateliers - Soudure

2.3.5 Atelier d'usinage



Figure 2-11 Ateliers - Usinage

2.3.6 Bancs et divers



Figure 2-12 Ateliers - Banc à rouleaux



Figure 2-13 Ateliers - Stock

3 Présentation du projet

3.1 Cahier des charges

3.1.1 Contenu

- Découverte du Campus Automobile et de sa finalité.
- Découverte du banc à rouleaux et types d'essais réalisés.
- Qu'est ce qu'un cycle de conduite normalisé ?
- Programmation d'un logiciel d'aide à la conduite d'un véhicule lors d'un essai normalisé au banc.

3.1.2 Description

- Le conducteur du véhicule doit avoir la possibilité de visualiser en temps réel sur un écran sa vitesse actuelle ainsi que celle de consigne (déterminée par le cycle prédéfini).
- Les points de passages des vitesses doivent également être visualisés, car imposés par le cycle (européen normalisé ou autre).
- Les programmes similaires existant sur le marché se présentent sous la forme d'un graphe déroulant verticalement affichant la vitesse consigne, avec la zone de marge d'erreur, et les points de passages des vitesses.
- Un curseur affiche sur ce graphe la vitesse réelle du véhicule. Le travail du pilote est alors de tenir le curseur dans la plage de tolérance et de passer les vitesses au bon moment.




3.2 Présentation du Projet

Le projet AutoCycle consiste en le développement d'un logiciel d'aide à la conduite pour un banc à rouleaux quatre roues motrices. L'application est placée sur un écran en face de l'opérateur et celui-ci doit suivre les informations indiquées par l'application.

Ce Projet contient un certain nombre de contraintes techniques et applicatives qui ont imposées un certain nombre de choix, lesquels sont détaillés dans les pages à venir.

Cette application est destinée à réaliser des cycles de conduite pour des voitures donc, la majorité des utilisateurs sont issus du domaine mécanique et non du domaine informatique. L'utilisateur final de l'application peut donc être n'importe quel type d'utilisateur, du novice au plus expérimenté.

Ceci impose un certains nombre d'aides à la compréhension du logiciel et à la facilité de manipulation.

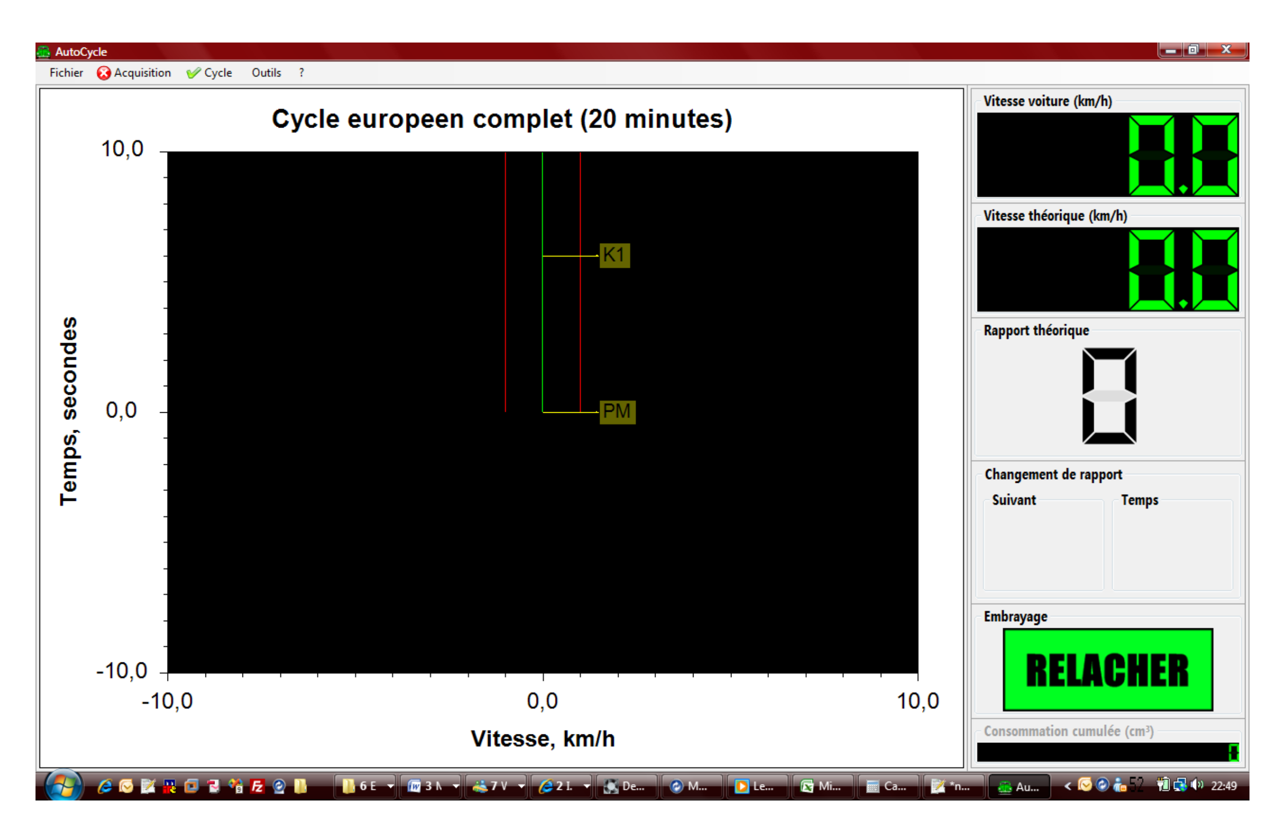


Figure 3-1 AutoCycle - Aperçu global

3.3 Matériel disponible

3.3.1 Banc AutoScan

« *Banc d'essai à rouleaux pour automobiles à deux ou quatre roues motrices conçu pour mesurer les performances d'un véhicule en conditions dynamiques.*

Destiné aux préparateurs automobiles et à l'enseignement technique, AutoScan est un outil objectif et rigoureux permettant aux utilisateurs de comparer et valider des choix techniques en phase de développement. » (Rotronics)



Figure 3-2 Ateliers - Banc à rouleaux

3.3.2 Boitier d'acquisition

Système embarqué réalisant l'acquisition des données directes du banc ainsi que de l'acquisition des données de la station météo. Celui-ci met les données à disposition sur le réseau via un câble RJ45 et le protocole MODBUS Ethernet TCP/IP ainsi que d'un protocole propriétaire de Rotronics.

Ports utilisés en mode connecté :

- 502 : protocole MODBUS.
- 503 : protocole propriétaire Rotronics.



Figure 3-3 Banc à rouleaux - Boitier d'acquisition



Figure 3-4 Banc à rouleaux - Boitier d'acquisition

3.3.3 Station météo

La station météo sert notamment à acquérir divers types de données météorologiques comme la température, pression, humidité, … mais également d'autres types de données du type fréquence, analogique, thermocouple.

Nous pouvons constater que selon le temps de réponse, certaines variables ne changeront pas au cours du test. Par exemple, le capteur de température à un temps de réponse de 15 minutes donc la température ne change que très peu au cours du test.

3.3.3.1 Capteurs

<u>Température ambiante</u>

- Etendue de mesure : 0 – 50 °C
- Précision : 1 °C
- Résolution : 0,1 °C
- Temps de réponse : 15 min

<u>Pression atmosphérique</u>

- Etendue de mesure : 750 – 1100 mbar
- Précision : 1,5 mbar
- Résolution : 0,1 mbar
- Temps de réponse : 1 min

<u>Entrées analogiques</u>

- Etendue de mesure : 0 – 15 V
- Précision : 0,03 V
- Résolution : 0,001 V
- Temps de réponse : 186 ms



Figure 3-5 Banc à rouleaux - Station météo

3.3.4 Gestion du banc

La gestion du banc est prise en main par un ordinateur dédié. Cet ordinateur est placé dans une armoire à serveur qui contient :

- Ordinateur complet,
- Imprimante,
- Module de richesse.

Le module de richesse ainsi que la prise diagnostique de la voiture rentrent via les ports série de l'ordinateur et servent à transmettre la valeur du capteur lambda ainsi que les diverses informations fournies par la prise diagnostique.

Le boitier d'acquisition du banc est quand à lui connecté via un câble RJ45 à l'ordinateur.

Notons qu'un deuxième écran est situé en face de la voiture et sert de clone au premier écran.



Figure 3-6 Banc à rouleaux - Affichage



Figure 3-7 Banc à rouleaux - Armoire pour ordinateur

3.3.5 Soufflerie

Un véhicule en fonctionnement intensif dégage beaucoup de chaleur qui est en général refroidie par le vent. Or, sur un banc à rouleaux, celle-ci n'est pas confrontée au vent et chauffe donc considérablement. C'est pourquoi il est nécessaire d'avoir une soufflerie permettant de refroidir le tout.

Cet ensemble de souffleries est composée d'une soufflerie principale ainsi que de deux souffleries plus petites et mobiles.



Figure 3-8 Banc à rouleaux - Soufflerie

3.3.6 Débitmètre

Le débitmètre est consacré au calcul de la consommation d'essence du véhicule.

Composé de deux parties, la première consiste au module au travers duquel passe l'essence. Celui-ci renvoie un nombre de pulsations par secondes correspondant à la consommation.

La deuxième partie est chargée de transformer ce nombre de pulsations en une consommation en termes de volume. C'est notamment ce module d'affichage/conversion qui, possédant une sortie analogique en volts, va se brancher sur la station météo afin que l'on en récupère les valeurs.



Figure 3-9 Débitmètre - module d'acquisition et d'affichage





3.3.7 Logiciel Kronos 4.0

Logiciel édité par la société Rotronics pour la gestion du banc à rouleaux ainsi que les essais classiques. Ce logiciel rassemble l'ensemble des données reçues via plusieurs types d'interface et permet d'agir en tant que serveur d'objets OPC pour ces données.

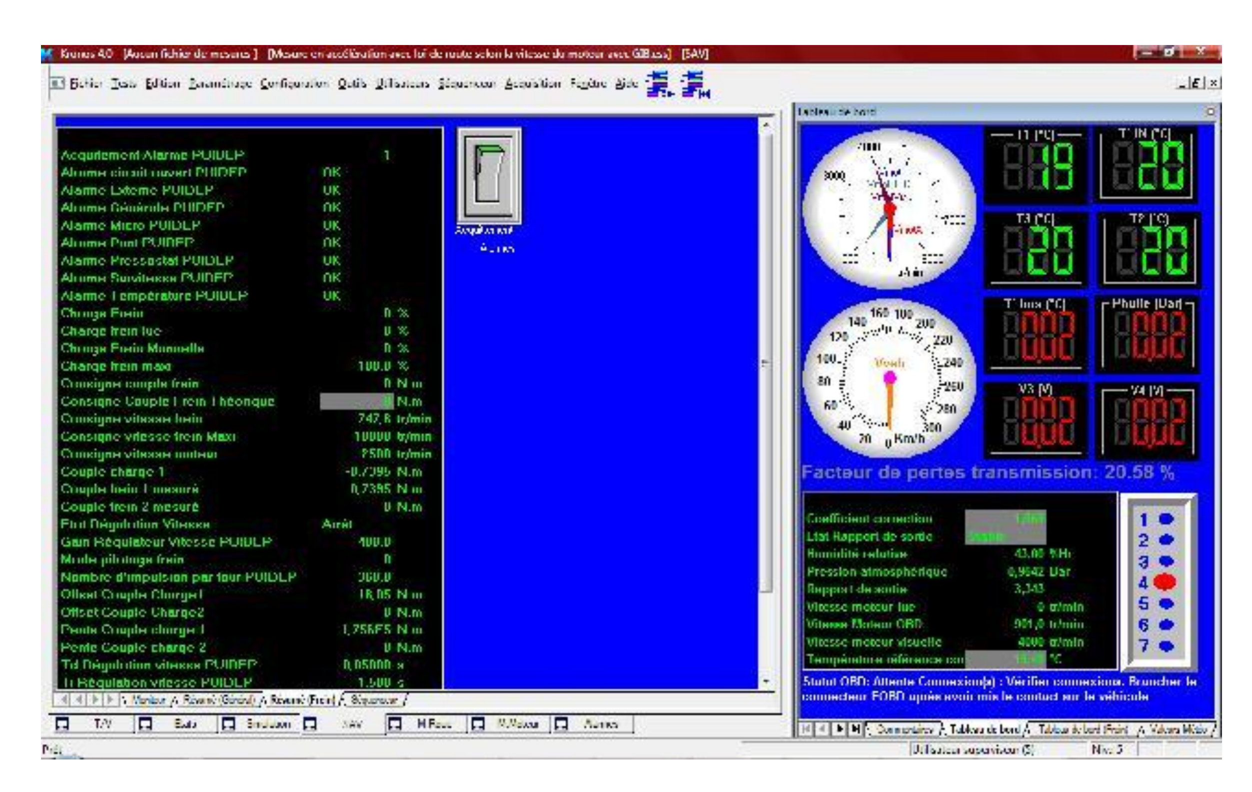


Figure 3-10 Kronos - Aperçu global

4 Description d'un cycle de conduite

Un cycle de conduite consiste en la simulation d'un trajet typique réalisé par une voiture. Il est représentatif des comportements de conduite réellement observés sur la route et sert à obtenir un tas d'informations propre à un véhicule pour les comparer avec les informations d'un autre véhicule.

Concrètement, il représente, sur un graphique, la vitesse du véhicule en fonction du temps. Il défini également les rapports de boite de vitesse, la vitesse, le statut de l'embrayage ainsi que diverses informations.

Généralement créé sous forme de norme par des organisations, pays, commissions, il sert à mesurer les performances d'un véhicule sur divers points tels que la consommation d'essence, le niveau d'émission de pollution, …

4.1 Utilité d'une norme

Pour réaliser des comparatifs, il est nécessaire que plusieurs conditions soient réunies :

- Effectuer les tests avec le même matériel (banc à rouleaux, logiciel, pc).
- Effectuer un cycle de conduite identique.
- Réussir le cycle de conduite avec un taux de réussite minimum.
- Mettre le véhicule dans des situations identiques. (coffre vide, aucun passager, …).

Sans ces diverses conditions, les tests seront faussés et ne pourront en aucun cas êtres comparés entre eux.

Ce standard est donc défini par des autorités et, dans le cas du NEDC, par l'Europe.

4.2 Nouveau Cycle de Conduite Européen (NEDC)

Dans le cadre de ce travail de fin d'études, on utilise le plus souvent le nouveau cycle de conduite européen (NEDC) consistant en une phase urbaine (maximum 50km/h) répétée 4 fois suivie d'une partie extra-urbaine (maximum 120km/h).

Le cycle NEDC, qui représente l'usage typique d'une voiture en Europe, à une durée de 19 minutes et 40 secondes.

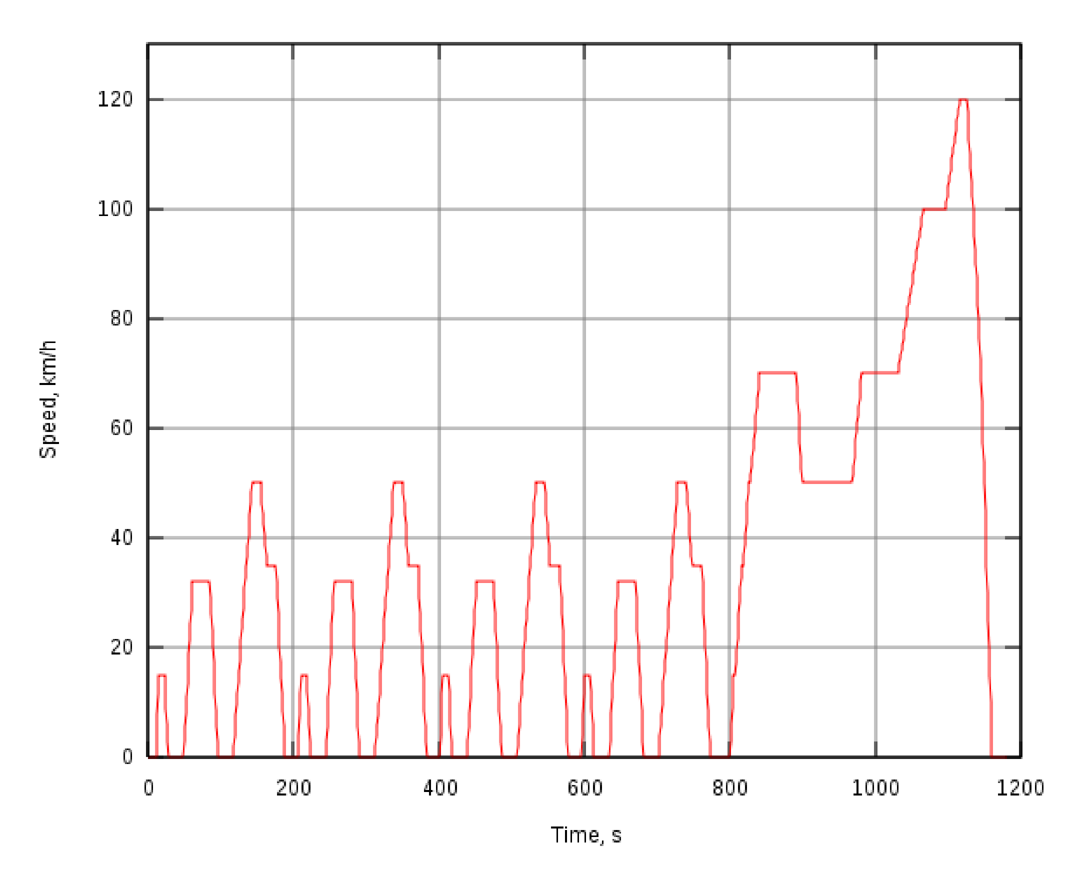


Figure 4-1 Cycle NEDC - Aperçu général

4.2.1 But

Le NEDC à pour objectif de mesurer le niveau d'émission de pollution de la voiture.

4.2.2 Tolérance

La norme prévoit une marge d'erreur pour la réalisation du cycle mais cependant, elle demeure très faible. Ce qui rend la réalisation du cycle extrêmement difficile et, demande un certain entrainement.

4.2.2.1 Vitesse

La tolérance autorisée est de +/- 2 km/h entre la vitesse indiquée et la vitesse théorique en accélération, en vitesse stabilisée, et en décélération.

4.2.2.2 Temps

La tolérance autorisée est de +/- 1 seconde entre le temps indiqué et le temps réel.

4.2.3 Contraintes de la norme

L'ensemble des contraintes imposées par la norme sont disponibles dans la norme européenne. Vu qu'il en existe des centaines, il serait inutile de les citer ci-dessous.

Toutefois, à titre d'exemple, voici une contrainte :

« *Le véhicule présenté doit être en bon état mécanique. Il doit être rodé et avoir parcouru au moins 3 000 km avant l'essai.* ». (Communautés Européennes, 2004)





4.2.4 Phases de conduites

4.2.4.1 Vue générale du cycle de conduite

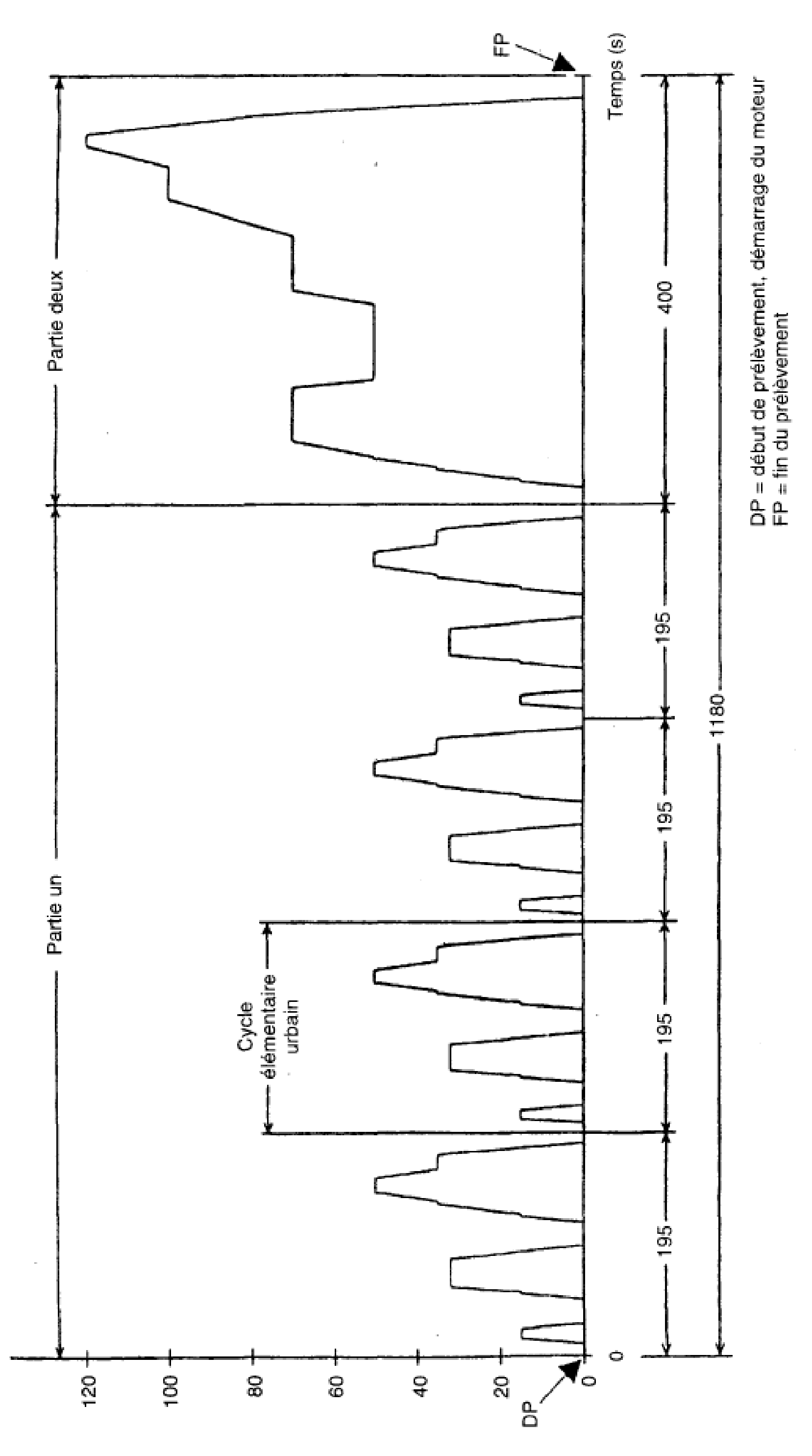


Figure 4-2 Cycle NEDC - Détails

4.2.4.2 Partie 1

La partie 1, d'une durée de 780 secondes, est représentée par un cycle de 195 secondes qui est réalisé quatre fois de suite. Pour la suite des points, nous allons détailler ce cycle.

4.2.4.2.1 Description

Type de trajet : *urbain*
Durée totale : *195 secondes*
Vitesse maximum : *50 km/h*
Vitesse moyenne : *19 km/h*
Distance théorique parcourue : *1,013km*

4.2.4.3 Décomposition par modes

	temps (secondes)	%
Ralenti	60	30,8
Ralenti, véhicule en marche, embrayage embrayé sur un rapport	9	4,6
Changements de vitesses	8	4,1
Accélérations	36	18,5
Marche à vitesse stabilisée	57	29,2
Décélérations	25	12,8
	195	**100**

4.2.4.4 Décomposition par vitesses

	temps (secondes)	%
Ralenti	60	30,8
Ralenti, véhicule en marche, embrayage embrayé sur un rapport	9	4,6
Changements de vitesses	8	4,1
Premier rapport	24	12,3
Deuxième rapport	53	27,2
Troisième rapport	41	21
	195	**100**

4.2.5 Détails des opérations

Cycle d'essai élémentaire urbain au banc à rouleaux (Partie 1)

Opération n°	Opération	Mode n°	Accélération (m/s²)	Vitesse (km/h)	Durée de chaque opération (s)	Durée de chaque mode (s)	Temps cumulé (s)	Rapport à utiliser dans le cas d'une boîte mécanique
1	Ralenti	1			11	11	11	6 s. PM + 5 s. K1 (¹)
2	Accélération	2	1,04	0-15	4	4	15	1
3	Vitesse stabilisée	3		15	8	8	23	1
4	Décélération	4	−0,69	15-10	2	5	25	1
5	Décélération, embrayage débrayé		−0,93	10-0	3		28	K1 (¹)
6	Ralenti	5			21	21	49	16 s. PM + 5 s. K1 (¹)
7	Accélération	6	0,83	0-15	5	12	54	1
8	Changement de vitesse			2	56			
9	Accélération		0,94	15-32	5		61	2
10	Vitesse stabilisée	7		32	24	24	85	2
11	Décélération	8	−0,76	32-10	8	11	93	2
12	Décélération, embrayage débrayé		−0,92	10-0	3		96	K2 (¹)
13	Ralenti	9			21	21	117	16 s. PM + 5 s. K1 (¹)
14	Accélération	10	0,83	0-15	5	26	122	1
15	Changement de vitesse			2	124			
16	Accélération		0,62	15-35	9		133	2
17	Changement de vitesse			2	135			
18	Accélération		0,52	35-50	8		143	3
19	Vitesse stabilisée	11		50	12	12	155	3
20	Décélération	12	−0,52	50-35	8	8	165	3
21	Vitesse stabilisée	13		35	13	13	176	3
22	Changement de vitesse	14			2	12	178	
23	Décélération							
24	Décélération		−0,87	► M19 35-10 ◄	7		185	2
	embrayage débrayé		−0,93	10-0	5		188	K2 (¹)
25	Ralenti	15			7	7	195	7 s. PM (¹)

(¹) PM: boîte au point mort, embrayage embrayé.
K1, K2: boîte sur le premier ou le deuxième rapport, embrayage débrayé.

Figure 4-3 Cycle NEDC - Opérations de la partie 1

4.2.6 Positionnement des opérations sur le graphique

Figure 4-4 Cycle NEDC - Position des opérations de la partie 1

4.2.6.1 Partie 2

4.2.6.1.1 Description

Type de trajet : *extra-urbain*
Durée totale : *400 secondes*
Vitesse maximum : *120 km/h*
Vitesse moyenne : *62,6 km/h*
Distance théorique parcourue : *6,955km*
Accélération maximale : *0,833m/s²*
Décélération maximale : *-1,389m/s²*

4.2.6.2 Décomposition par modes

	temps (secondes)	%
Ralenti	20	5
Ralenti, véhicule en marche, embrayage embrayé sur un rapport	20	5
Changements de vitesses	6	1,5
Accélérations	103	25,8
Marche à vitesse stabilisée	209	52,2
Décélérations	42	10,5
	400	**100**

4.2.6.3 Décomposition par vitesses

	temps (secondes)	%
Ralenti	20	5
Ralenti, véhicule en marche, embrayage embrayé sur un rapport	20	5
Changements de vitesses	6	1,5
Premier rapport	5	1,3
Deuxième rapport	9	2,2
Troisième rapport	8	2
Quatrième rapport	99	24,8
Cinquième rapport	233	58,2
	400	**100**




4.2.7 Détails des opérations

Opération n°	Opération	Mode n°	Accélération (m/s^2)	Vitesse (km/h)	Durée de chaque opération (s)	Durée de chaque mode (s)	Temps cumulé (s)	Rapport à utiliser dans le cas d'une boîte mécanique
1	Ralenti	1			20	20	20	K1 ([1])
2	Accélération		0,83	0-15	5		25	1
3	Changement de vitesse				2		27	-
4	Accélération		0,62	15-35	9		36	2
5	Changement de vitesse	2			2	41	38	-
6	Accélération		0,52	35-50	8		46	3
7	Changement de vitesse				2		48	-
8	Accélération		0,43	50-70	13		61	4
9	Vitesse stabilisée	3		70	50	50	111	5
10	Décélération	4	−0,69	70-50	8	8	119	4 s. 5 + 4 s. 4
11	Vitesse stabilisée	5		50	69	69	188	4
12	Accélération	6	0,43	50-70	13	13	201	4
13	Vitesse stabilisée	7		70	50	50	251	5
14	Accélération	8	0,24	70-100	35	35	286	5
15	Vitesse stabilisée	9		100	30	30	316	5 ([2])
16	Accélération	10	0,28	100-120	20	20	336	5 ([2])
17	Vitesse stabilisée	11		120	10	10	346	5 ([2])
18	Décélération		−0,69	120-80	16		362	5 ([2])
19	Décélération		−1,04	80-50	8		370	5 ([2])
20	Décélération, embrayage débrayé	12	−1,39	50-0	10	34	380	K5 ([1])
21	Ralenti	13			20	20	400	PM ([1])

([1]) PM: boîte au point mort, embrayage embrayé.
K1, K5: boîte sur le premier ou le cinquième rapport, embrayage débrayé.

([2]) Si le véhicule est équipé d'une boîte de vitesses de plus de cinq rapports, les rapports supplémentaires pourront être utilisés en accord avec les recommandations du constructeur.

Figure 4-5 Cycle NEDC - Opérations de la partie 2

4.2.8 Positionnement des opérations sur le graphique

Figure 4-6 Cycle NEDC - Position des opérations de la partie 2

5 Analyse de l'application

5.1 Etude de faisabilité

Dans une certaine logique de conception, il est d'abord primordial de définir l'aspect global du projet et de finir avec la solution matérielle. L'expression populaire visant à caractériser cela est le fait de prendre le problème par au dessus.

Dans notre cas, ce n'était pas la meilleure solution à appliquer. En effet, j'ai été limité dans la conception par le matériel déjà existant. Il était donc inutile de chercher à créer une quelconque solution qu'il aurait fallu adapter par la suite en fonction des restrictions matérielles.

C'est pourquoi, avant d'élaborer toute ébauche de projet, il est important de voir les diverses possibilités qui s'offrent à nous en terme de développement en fonction des contraintes imposées. Les choix technologiques concernant les outils utilisés pour le développement sont expliqués au point 5.2.

5.1.1 Acquisition des capteurs

Le banc est connecté à l'ordinateur via un câble RJ45 de type Ethernet.

Après quelques recherches infructueuses dans la documentation du banc, j'ai réalisé des tests réseaux pour voir que la communication est de type TCP/IP et utilise deux ports.

J'ai donc téléphoné à la société responsable du banc pour expliquer mon problème et obtenir des renseignements. Voila ce qui en découle :

- La communication TCP/IP se fait via le protocole MODBUS ainsi que un protocole propriétaire de Rotronics dont ils ne veulent pas fournir la documentation.
  - ⇨ ***Protocole inconnu: impossibilité d'établir une connexion au banc.***
  - ⇨ ***Protocole TCP/IP: impossible de connecter plus d'un logiciel sur le banc. Or, nous avons besoin de faire tourner Kronos V4.0 et AutoCycle V1.0.***

- Le logiciel Kronos V4.0 agit en tant que serveur OPC pour redistribuer l'ensemble des valeurs des capteurs.
  - ⇨ ***Obligation d'utiliser Kronos V4.0 comme passerelle pour communiquer avec le banc***
  - ⇨ ***Fréquence d'actualisation imposée par Kronos V4.0***

Face à l'ensemble de ces contraintes et le peu de choix proposé, la seule solution possible est l'utilisation d'objets OPC et donc, de passer par le logiciel Kronos V4.0 pour obtenir les valeurs désirées.

5.1.2 Rendu graphique des résultats

Devant le problème de la représentation graphique, il a fallu trouver une solution qui permettait d'offrir le plus d'options au programmeur, tant au niveau du nombre de fonctions existantes, de la facilité d'utiliser celles-ci, mais aussi du rendu désiré.

Après de nombreuses recherches, quelques éléments en sont ressortis :

- ➔ En JAVA, je n'ai trouvé qu'une librairie graphique remplissant l'ensemble des fonctions requises excepté le fait qu'elle soit prévue pour une utilisation en temps réel
- ➔ En C#, beaucoup de librairies graphiques gratuites existent mais une seule permet une certaine souplesse au niveau de la manipulation de graphique et qui plus est, permet l'utilisation du rendu en temps réel.

Cette librairie graphique en C#, c'est la librairie ZedGraph qui est détaillée au paragraphe 5.2.5.

5.1.3 Conclusion

Face a ces deux contraintes obligatoires pour lesquelles une solution a été trouvée, le projet est donc viable. Celui-ci peut donc commencer à être pensé et conçu.

Le chapitre suivant sera donc consacré à l'analyse et la conception du logiciel.

5.2 Outils

5.2.1 Choix du langage de développement

Le programme étant essentiellement basé sur des fenêtres et des communications via des objets représentant des classes, je me suis porté vers un langage orienté objet et qui plus est, de haut niveau.

Contraintes :

- Une librairie graphique temps réel,
- Une librairie de communication avec un serveur OPC.

N'ayant pas trouvé de solution gratuite à ces deux contraintes dans le langage JAVA, je me suis orienté vers le langage C#.NET où j'ai pu trouver ces deux éléments beaucoup plus aisément.

5.2.2 Plateforme Microsoft .NET

Cette plateforme est ensemble logiciel développé par la société Microsoft pour rendre les applications portable et facilement accessible depuis Internet ainsi que de grandement augmenter la facilité de développement. Elle est limitée à l'environnement Microsoft Windows.

5.2.2.1 Architecture

La principale caractéristique de .NET est de passer par un code intermédiaire, appelé le *Common Intermédiate Language*. De ce fait, le compilateur .net va compiler le langage (c#.NET, asp.NET, …) en un même langage intermédiaire. Par la suite, c'est le *Common Language Runtime* qui va être chargé de traduire ce code intermédiaire en code machine. De cette manière, on assure une indépendance au niveau de la plateforme, tant bien en développement qu'en exécution.

L'architecture comprend notamment des outils de sécurité, interopérabilité (utilisé dans ce TFE), gestion des erreurs et gestion de la mémoire.

Un grand avantage de la gestion mémoire, par exemple, est de ne plus utiliser les pointeurs comme en C/C++. Le programmeur n'as plus à se soucier d'allouer et de libérer de la mémoire, c'est l'architecture .NET qui s'en occupe automatiquement.

L'ensemble de cette architecture repose d'une manière globale dans la *Common Langage Infrastructure*, indépendante du langage utilisé.

5.2.2.2 Schéma

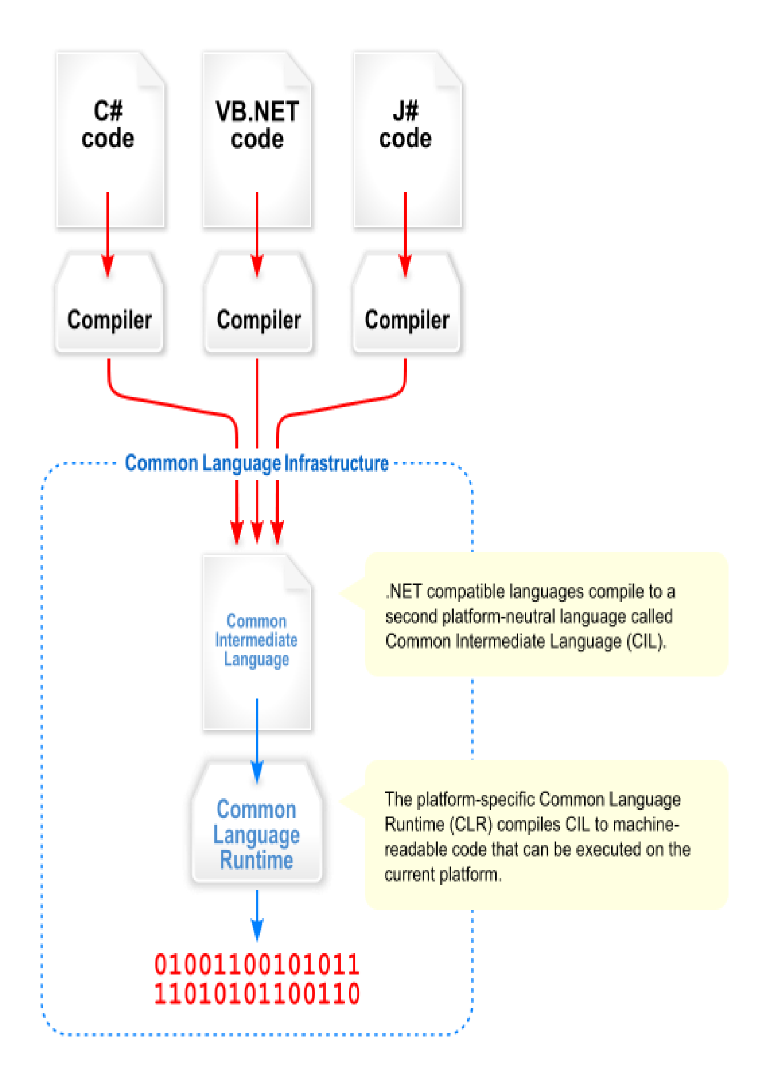


Figure 5-1 Framework .NET - Schéma de fonctionnement

5.2.3 Framework .NET

Sous ensemble de la plateforme Microsoft .NET, le Framework .NET est un ensemble de bibliothèques agissant comme des briques de bases pour le développement d'applications.

Ces bibliothèques sont tenus à des règles spécifiques et à une certaine rigueur, de sorte de pouvoir les faire interagir entre-elles et de faciliter la maintenance, gestion.

Une version plus légère et limitée comportant le nom de .NET Compact Framework est apparue avec la version 5 du Framework. Celle-ci permet notamment l'exécution sur un système d'exploitation Microsoft Windows Mobile.

Le choix s'est porté sur la version 3.0 du Framework .NET vu que les nouveaux éléments de la version 3.5 ne sont pas utilisés.





5.2.3.1 Evolution du Framework .NET

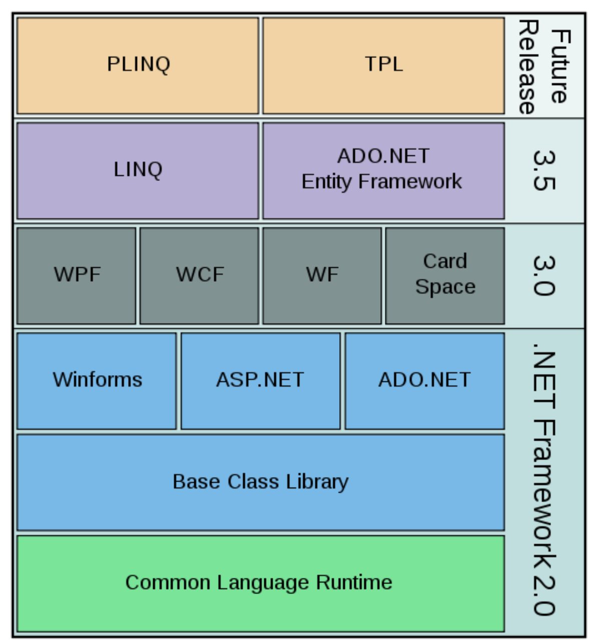


Figure 5-2 Framework .NET - Pile des composants

5.2.4 Microsoft Visual Studio 2008

Pour coder un logiciel requérant la manipulation de nombreux objets et interface graphique, il devient vite compliqué de tout gérer dans un bloc notes. C'est pourquoi j'utilise le logiciel Microsoft Visual Studio 2008 qui me facilite la gestion du projet, des classes fenêtres ainsi que des composants réutilisables.

5.2.5 Librairie graphique ZedGraph

ZedGraph est une librairie graphique open-source destinée aux environnements de développement Microsoft .NET. Elle permet une grande souplesse de création, modification et permet de représenter presque n'importe quel type de graphe. Elle est, prévue pour faire des graphiques en temps réel.

Cependant, elle utilise l'API *GDI+* pour le rendu du graphique. Or, cette API effectue ses calculs au sein du CPU et non du GPU. Ce qui occasionnera une très grosse charge du CPU pendant le déroulement du graphique.

5.2.5.1 Quelques exemples

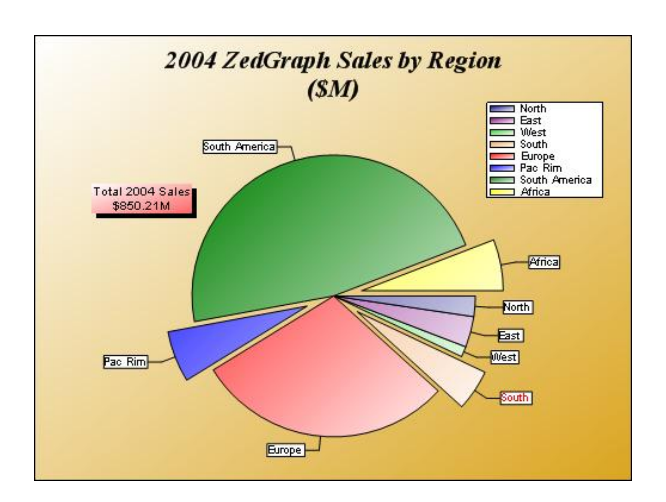


Figure 5-3 ZedGraph - Exemple 1

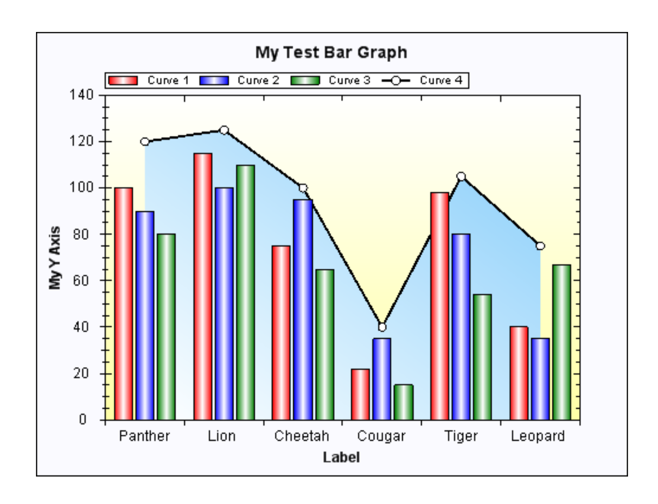


Figure 5-4 ZedGraph - Exemple 2

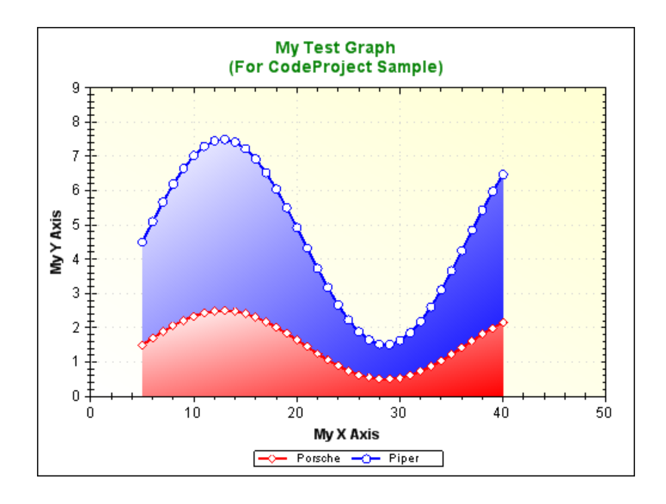


Figure 5-5 ZedGraph - Exemple 3

5.2.6 Librairie OPCdotNETLib

Le problème majeur dans les communications OPC en .NET est qu'il n'existe pratiquement aucune solution gratuite. La solution de la fondation OPC consiste en une librairie .NET pour OPC mais disponible uniquement pour les membres de la fondation. Or, cette inscription annuelle s'élève à 2000 euros par an. Ce qui est, dans le cadre de ce travail, complètement démesuré.

Après des recherches approfondies, j'ai finalement trouvé UNE solution quand au problème de la communication.

Cette solution, c'est la librairie OPCdotNETLib créée par VISCOM .NET Team.
Cette équipe à pris une librairie de connexion OPC codée en C++ et l'a recodée entièrement en code managé C#.

Elle contient tout ce dont j'ai besoin et, grâce à l'aide de l'auteur qui continue à répondre sur le forum de sa librairie depuis des années, d'autres fonctions que j'ai modifié.

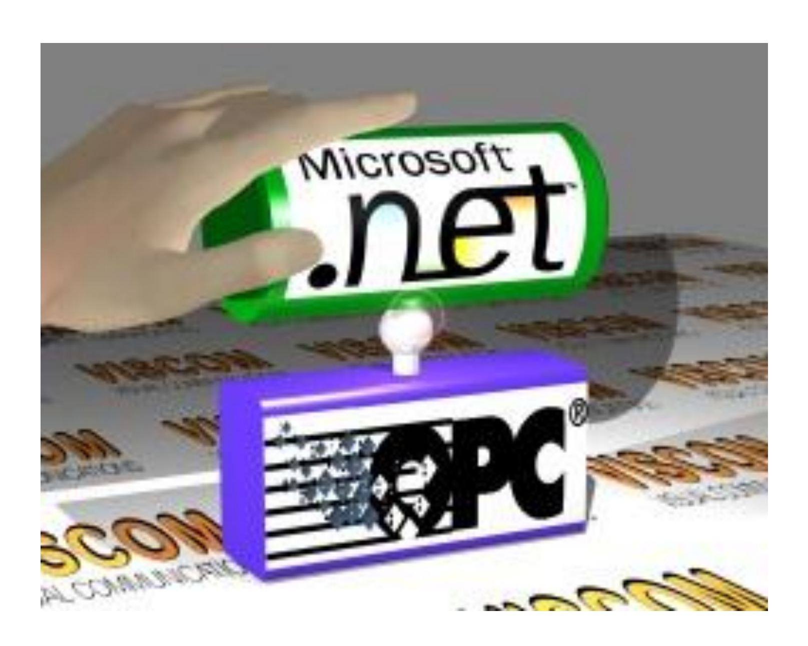


Figure 5-6 VISCOM - Symbolique de la librairie OPCdotNETLib





5.2.7 Librairie Owf.Controls.DigitalDisplayControl

Voulant posséder un affichage digital du même type que le logiciel déjà existant sur le banc (Figure 5-8), j'ai choisi cette librairie graphique permettant d'afficher un compteur digital (Figure 5-7).

Je l'ai Modifié (Figure 5-9) pour l'adapter à mon programme.



Figure 5-7 Affichage digital - Exemple



Figure 5-8 Affichage digital - Aspect recherché

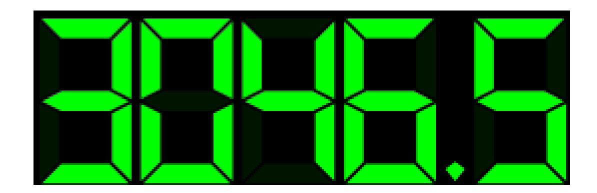


Figure 5-9 Affichage digital - Aspect obtenu

5.2.8 Librairie ExcelPackage

L'adoption par Microsoft du standard Office Open XML pour sa suite bureautique Microsoft Office 2007 à permis un bond en avant dans le développement et la manipulation de ses fichiers.

Avant la version 2007, Microsoft utilisait son propre format de fichier propriétaire, ce qui avait pour conséquence que personne ne pouvait manipuler aisément ces fichiers.

Avec l'apparition du standard Office Open XML, les formats sont dit « ouverts ». C'est-à-dire que n'importe qui peut en lire le contenu et les manipuler. De plus, pour en faciliter la manipulation, ils sont basés sur le langage XML qui en simplifie la rédaction à l'aide d'un système de balisage.

Pour nous, programmeurs, ceci est un avantage considérable car cela nous permet d'exploiter ce type de document. C'est grâce à l'utilisation de ce standard que les fichiers de rapports seront générés au format .XSLX (Excel open XML).

Quand à la librairie ExcelPackage, elle exploite notamment cette norme à l'aide de classes et parseurs situés dans l'api .NET 3.0 System.IO.Packaging. Ce package fourni des méthodes pour faciliter la manipulation des dit fichier.



Figure 5-10 ExcelPackage - Logo de la librairie

5.3 Analyse

Le logiciel doit :

- se connecter à un serveur OPC et acquérir un ensemble de valeurs.
- afficher ces valeurs pour l'utilisateur.
- ouvrir des fichiers CSV de cycles.
- afficher ces cycles de manière graphique.
- permettre la réalisation du cycle décrit dans le fichier CSV en temps réel.
- générer un rapport concernant le cycle réalisé à la fin de celui-ci ou pendant l'essai
- gérer la totalité des erreurs pouvant subvenir afin de ne pas faire planter le programme.
- pouvoir tracer des évènements.
- être multilingue.
- être le plus personnalisable possible.

5.3.1 Organisation de la solution

Le premier objectif de l'organisation du projet est de séparer au mieux l'ensemble des classes / modules qui seront utilisés. Ainsi, pour ne pas avoir un unique projet qui serait surchargé par toutes les classes, une solution appelée « TFE » contient un ensemble de projets qui ont pour fonction des taches bien distinctes.

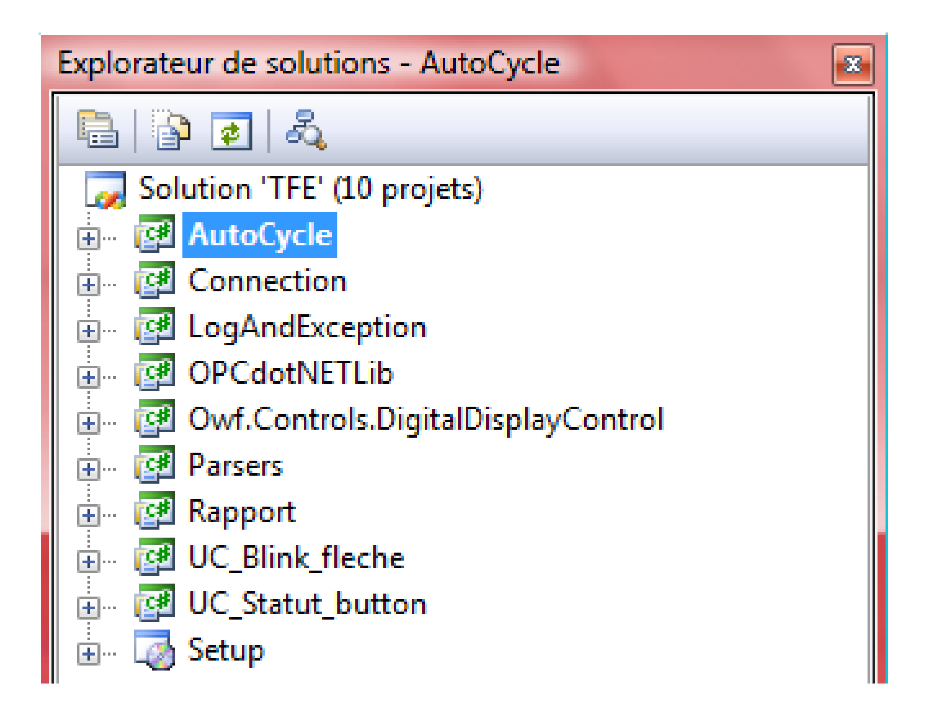


Figure 5-11 AutoCycle - Arborescence de la solution

Deuxièmement, pour faciliter le développement futur, ces projets sont majoritairement indépendant les uns des autres. Excepté cependant une classe de log qui se retrouve dans la plus part des projets. Ces dépendances sont détaillées dans chaque section consacrée au projet respectif.

Et dernièrement, pour aider à la compréhension du fonctionnement global du programme, celui-ci est conçu sous forme de couches. Ainsi, l'interface utilisateur ne peut pas communiquer directement avec la classe de communication mais doit passer par un intermédiaire.

Cet intermédiaire est la classe « BACK.cs » ou, plus communément, l'épine dorsale du programme qui est chargée d'instancier l'ensemble des projets ainsi que les communications.

5.3.2 Détails des projets attendus

Interface utilisateur

Projet principal englobant l'épine dorsale du programme (point de démarrage) ainsi que l'ensemble des interfaces utilisateurs.

Conservation des logs et affichage des messages d'erreurs

Un système d'archivage d'évènements dans un fichier journalier en vue d'aider le développeur quand au traitement d'erreurs et d'informations. Ce système sert également à afficher les messages d'erreurs et d'informations à l'utilisateur. Chaque projet désirant effectuer un de ces trois points devra découler de ce projet.

Manipulation de fichiers CSV

Projet permettant l'ouverture de fichiers CSV et de l'insertion de ses éléments dans des listes de listes, correspondant respectivement aux lignes et aux champs de ces lignes. Une fonction permettant l'écriture de fichiers CSV est prévue mais non utilisée.

Connexion

Projet de connexion au serveur OPC qui dispose des librairies nécessaires ainsi que d'une configuration de la connexion via un fichier XML.

Contrôles d'utilisateurs

Un ensemble de projets comprenant divers modules d'affichages de l'interface utilisateur tels que le statut de l'embrayage, la prévention de changement de rapport via une flèche clignotante ou encore un affichage de type digital.

Module d'installation

Le programme dispose d'un fichier d'installation qui devra notamment créer un dossier d'installation, un raccourci sur le bureau ainsi que une entrée dans le dossier « programmes » du menu « démarrer ».

5.3.3 Respect du cahier des charges

- Le conducteur du véhicule doit avoir la possibilité de visualiser en temps réel sur un écran sa vitesse actuelle ainsi que celle de consigne (déterminée par le cycle prédéfini).
    - ➢ *Utilisation d'un module d'affichage digital de la vitesse réelle ainsi que pour la vitesse théorique qui elle, est calculée au moyen d'une formule d'accélération.*

- Les points de passages des vitesses doivent également être visualisés, car imposés par le cycle (européen normalisé ou autre).
    - ➢ *Création d'un module d'affichage « UC_Blink_Fleche » chargé de prévenir l'utilisateur d'un futur changement de rapport grâce à un décompte en seconde ainsi qu'une flèche indiquant si l'utilisateur doit augmenter ou rétrograder le rapport.*
    - ➢ *Insertion d'étiquettes sur le graphique indiquant à l'utilisateur qu'il doit changer de rapport et indiquant également quel est le rapport courant.*

- Les programmes similaires existant sur le marché se présentent sous la forme d'un graphe déroulant verticalement affichant la vitesse consigne, avec la zone de marge d'erreur, et les points de passages des vitesses.
    - ➢ *Le graphique déroulant est représenté verticalement.*
    - ➢ *Les marges d'erreurs sont également visualisables sur le graphique.*
    - ➢ *La vitesse de consigne est affichée grâce au module d'affichage digital expliqué un peu plus haut.*

- Un curseur affiche sur ce graphe la vitesse réelle du véhicule. Le travail du pilote est alors de tenir le curseur dans la plage de tolérance et de passer les vitesses au bon moment.
    - ➢ *Le curseur n'est pas repris en tant que tel mais est le dernier point (dernière mesure effectuée) de la courbe de la vitesse réelle de la voiture. Grâce aux options, il est possible de centrer le graphique sur ce dernier point ce qui en fait un curseur positionné au milieu du graphique.*

6 Conception de l'application

6.1 Interaction des projets

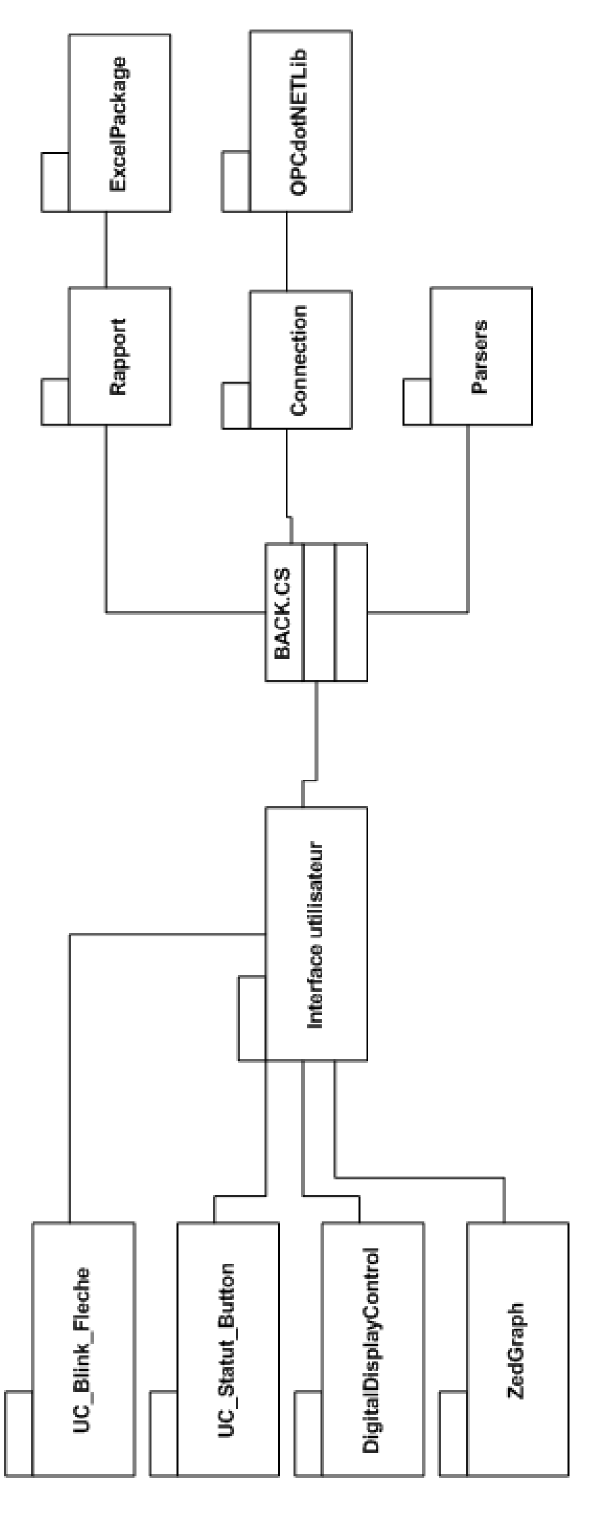


Figure 6-1 AutoCycle - Interaction des projets

Comme dit précédemment lors de l'analyse, les modules ne communiquent pas directement entre eux mais passent par la classe intermédiaire « BACK.cs ». Le point d'entrée du programme est situé dans cette classe qui est également l'épine dorsale, c'est elle donc qui instancie l'ensemble des projets dans ses variables privées et permet de manipuler ces différents objets.

Remarques à propos de la Figure 6-1 pour un souci de lisibilité :

- *les noms des packages ne sont pas les noms exacts tels que décrit dans l'organisation de la solution du sous chapitre « Analyse ».*
- *Le package de log n'est pas représenté car la majorité des projets en dérivent directement.*
- *La classe « BACK.cs » fait partie du projet « AutoCycle » qui lui-même est chargé de représenter l'interface utilisateur. Cependant, pour mieux discerner les communications inter-projets, nous avons dissociés cette classe du reste du projet. Notez toutefois que l'on aurait très bien pu créer un projet qui ne contiendrait que la classe « BACK.cs ».*

La suite de ce chapitre explique l'ensemble détaillé des projets ainsi que des problèmes rencontrés pour chacun d'entre eux et les solutions techniques apportées.

6.2 Projet « AutoCycle »

Projet principal destiné à coordonner l'ensemble des autres projets au travers de la classe « BACK.cs » ainsi que de s'occuper de l'interface utilisateur.

Dépendances spécifiques

- Connexion
- LogAndException
- Owf.Controls.DigitalDisplayControl
- Parsers
- Rapport
- UC_Blink_fleche
- UC_Statut_button
- ZedGraph

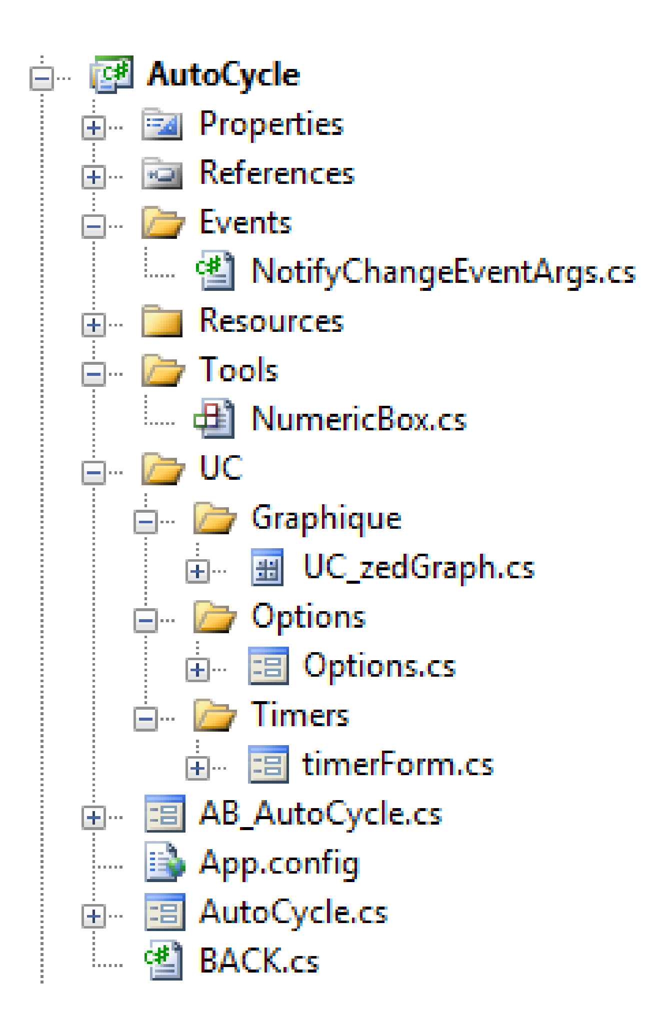


Figure 6-2 AutoCycle - Arborescence du projet "AutoCycle"

6.2.1 AutoCycle.cs

Interface principale de l'utilisateur interagissant avec l'ensemble des modules listés ci-dessous.

6.2.2 BACK.cs

Classe principale du programme, celle-ci agit en tant qu'épine dorsale sur laquelle est connectée l'ensemble des autres objets. Elle est chargée de faire la liaison entre les modules du programme. Pour preuve, il suffit de voir les différents échanges au sein des modules pour se rendre compte que cette classe est le point central.

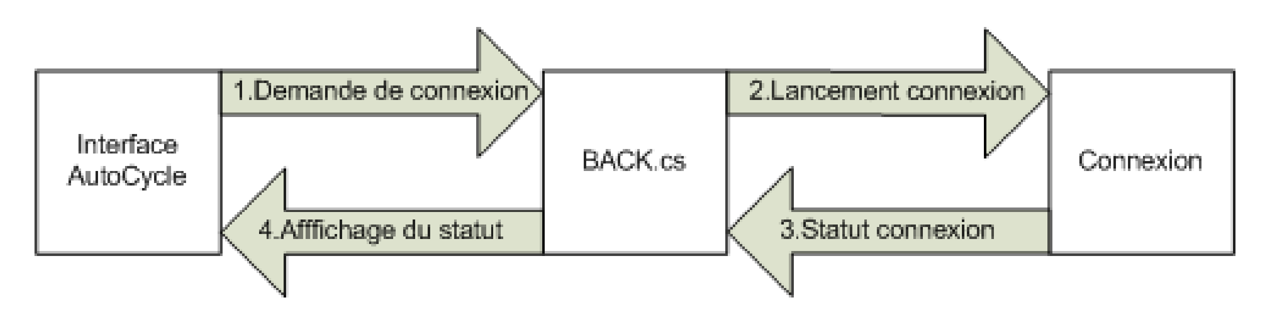


Figure 6-3 Communication - Connexion au serveur OPC

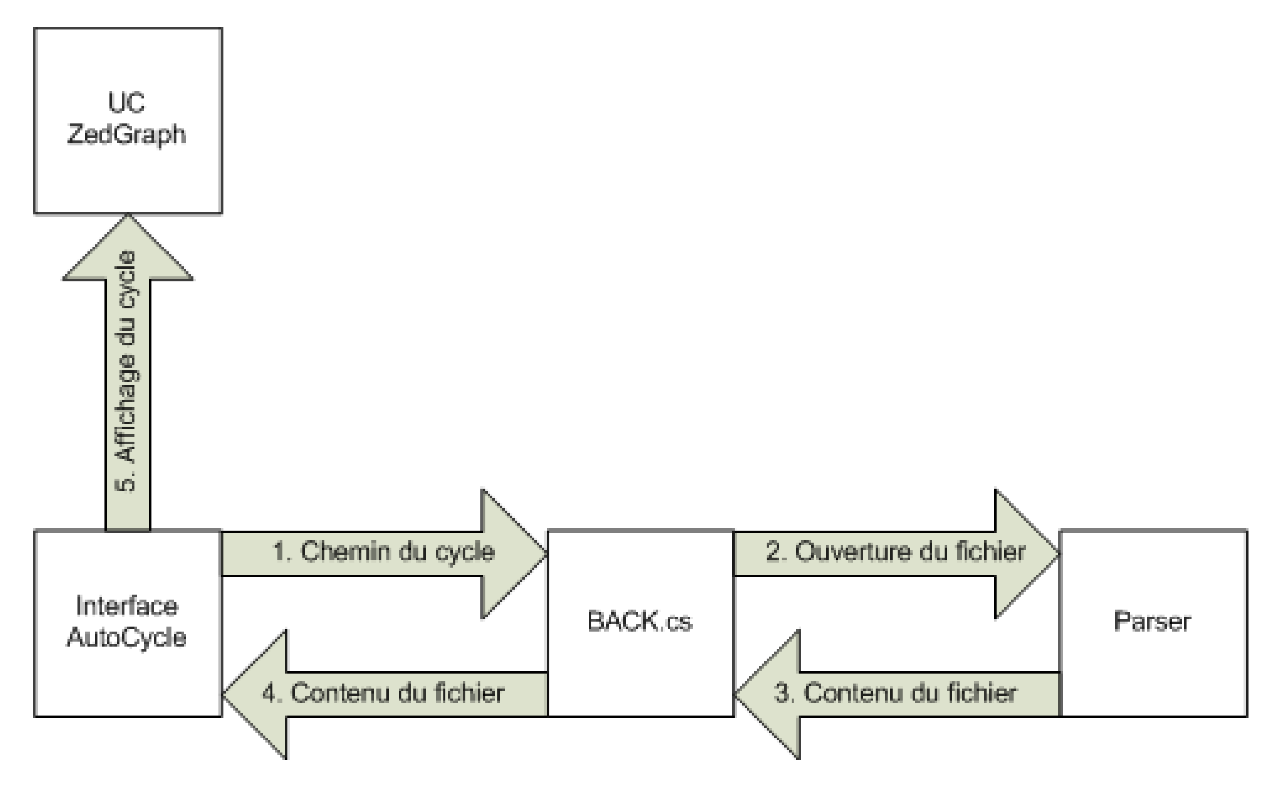


Figure 6-4 Communication - Ouverture d'un fichier de cycle

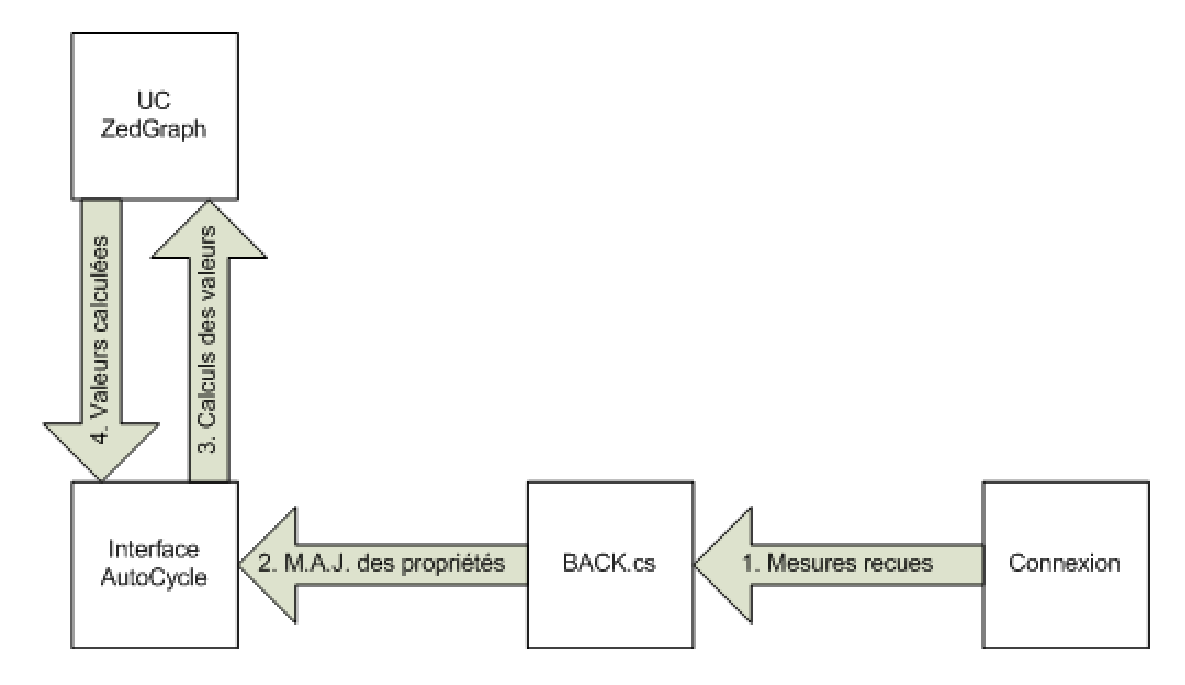


Figure 6-5 Communication - Récupération des données

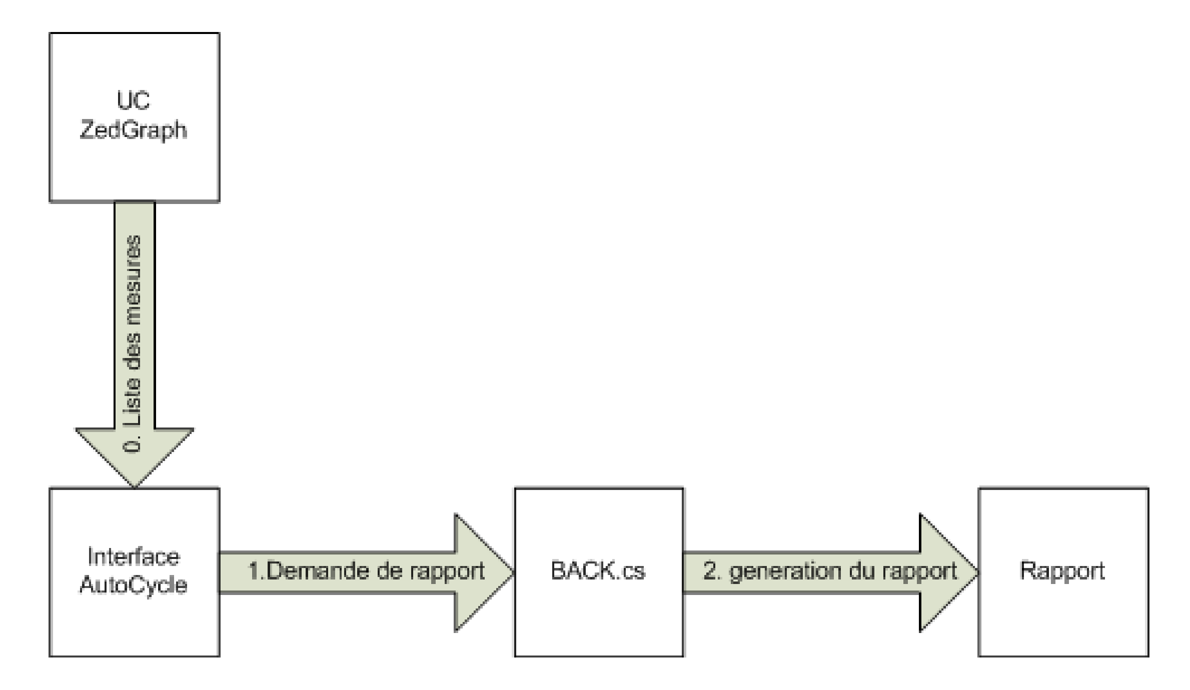


Figure 6-6 Communication - Génération de rapport

6.2.3 timerForm.cs

Contrôle utilisateur qui affiche un décompte dans une fenêtre. Celle-ci ne possède pas de bouton pour la fermeture ou la minimiser/maximiser. Elle apparait lors du lancement du cycle et a pour but de laisser le temps à l'opérateur de s'installer dans le véhicule avant le début du cycle.

Les valeurs autorisées sont comprises entre 0 et 999.

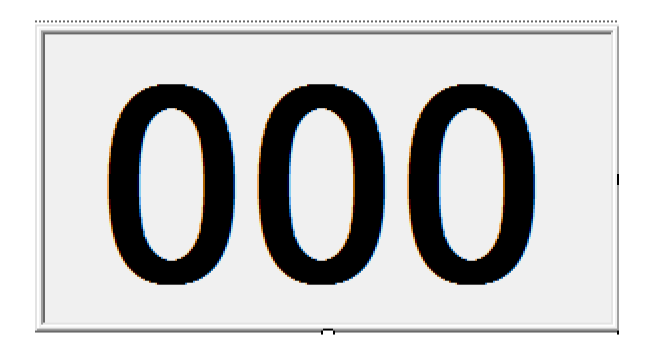


Figure 6-7 Timer - Fenêtre de minuterie

6.2.4 Options.cs

Boite à outils permettant de personnaliser l'application à de nombreux niveau. L'utilisateur pourra notamment choisir la langue, les fréquences d'acquisition, les couleurs, les délais, les paramétrages du graphique et pour terminer, les chemins par défaut des dossiers et fichiers.

La modification d'une de ces valeurs entraine l'utilisation de l'ApplicationBinding expliqué au point 6.9.3. Ainsi, chaque modification est répercutée sur l'ensemble du programme et ce, en temps réel.

6.2.5 UC_zedGraph.cs

Elément fondamental afin de faire fonctionner le programme, ce contrôle utilisateur est responsable de nombreuses fonctionnalités liées au graphique et au calcul de données.

Le graphique est démarré à l'aide d'un timer qui va effectuer un certain nombre de taches périodiquement. Mais pour mieux comprendre ce point, il est plus intéressant de voir le code commenté ci-dessous.





6.2.5.1 Fonctionnement lors d'un tick du timer

```
// Make sure that the curvelist has at least one curve
if (zedGraphControlVitesse.GraphPane.CurveList.Count <= 0)
    return;

// Get the first CurveItem in the graph
LineItem curve = zedGraphControlVitesse.GraphPane.CurveList[0] as LineItem;
if (curve == null)
    return;

// Get the PointPairList
IPointListEdit list = curve.Points as IPointListEdit;
// If this is null, it means the reference at curve.Points does not
// support IPointListEdit, so we won't be able to modify it
if (list == null)
    return;

// Temps mesuré en secondes
double time = (Environment.TickCount - _tickStart) / 1000.0;

// Vérification de fin de cycle
if ((double)_arret <= time)
{
    StopCycle();
    initAffichages();

    NotifyChangeEventArgs es = new NotifyChangeEventArgs(1,1);
    OnEndOfCycle(this, es);

    return;
}

// Ajout du point de la vitesse réelle au temps donné
list.Add((double)_vitesse, (double)time);

// Centrage des axes
rescaleAxesCenter(false);

// Modifier la position courante dans la List<List<Object>> du cyle
CurrentPositionList = getCurrentPositionList();

// Méthodes pour les modifications d'affichages sur la fenetre principale
setVitesseTheorique(time);
setEmbrayage(time);
setRapport(time);
setRapportNext(time);

// Vérifier les prochaines étiquettes
ModifierEtiquettes();

// Ajout des valeurs des capteurs pour le rapport
List<double> a = new List<double>();
a.Add(time);
a.Add(VitesseTheorique);
a.Add(Vitesse);
a.Add(Lambda);
a.Add(DebitAir);
a.Add(Consommation);
a.Add(TemperatureAmbiante);
a.Add(PressionAmbiante);
a.Add(checkSpeedOutOfRange(getVitesseTheoriqueAccelerationMIN(time),Vitesse,
getVitesseTheoriqueAccelerationMAX(time)));

RapportCapteurs.Add(a);

// Re-dessiner
zedGraphControlVitesse.Invalidate();
```




Explication

1. Vérifier que le graphique possède au moins la courbe de vitesse réelle.
2. Obtenir la référence vers la courbe de vitesse réelle.
3. Obtenir la liste de point de cette courbe.
4. Obtenir le temps écoulé depuis le début du cycle.
5. Vérifier si le cycle est terminé.
6. Ajout de la nouvelle mesure a la courbe.
7. Centrage du graphique sur la nouvelle mesure.
8. Modifier la valeur indiquant la position courante dans la liste de points en fonction du temps.
9. Modifier les valeurs d'affichage de la fenêtre AutoCycle (embrayage, rapport, …).
10. Vérifier les prochaines étiquettes à afficher et supprimer celles déjà dépassées.
11. Ajout de l'ensemble des mesures prises dans une liste en vue de créer le rapport.
12. Rafraichissement du graphique.

6.2.5.2 Calcul de la vitesse théorique

La vitesse théorique n'est disponible par défaut qu'une fois par seconde. Cela ne pose pas de problème dans le cas où le cycle est dans un palier de vitesse unique. Cependant, lors des courbes d'accélération et de décélération il est impossible de savoir la vitesse théorique.

Pour ce faire, j'utilise la formule de physique portant sur l'accélération. Celle-ci dit que, en connaissant le point précédent et le point suivant, il est possible de connaitre l'accélération effectuée.

Une fois que l'on possède cette accélération ainsi que le temps actuel, il est aisé de retrouver la vitesse théorique actuelle.

$$a = \frac{v'' - v'}{t}$$

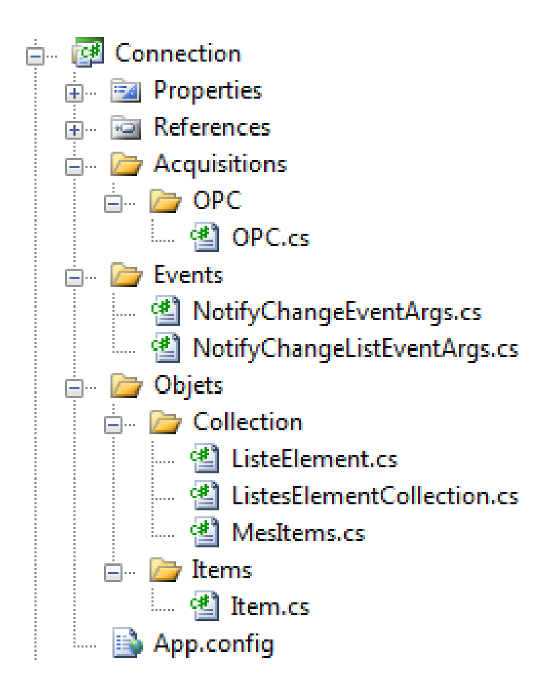
a = accélération (positive ou négative) en m / s²
v'' = vitesse à l'instant final en m / s
v' = vitesse à l'instant initial en m / s
v'' - v' = variation de vitesse en m / s
t = temps en s

6.3 Projet « Connexion »

Projet destiné à effectuer la lecture d'Items sur le serveur OPC.

<u>Dépendances spécifiques</u>

- LogAndException
- OPCdotNETLib



6.3.1 Paramétrage de la connexion

Afin de rendre le projet le plus générique possible, il est nécessaire d'utiliser un fichier de configuration pour la connexion ainsi que l'ensemble des éléments qui doivent être lus. La question du choix du type de fichier s'est posée : dois-je utiliser un fichier XML, un fichier de type «HashTable », un fichier de configuration ou encore un autre ?

Pour des soucis de facilité, l'option du fichier de configuration à été retenue mais bien vite un problème à été rencontré : comment créer une collection d'ensembles de valeurs dans ce fichier ?

C'est donc avec l'idée de réaliser un fichier de propriétés à la manière d'un fichier XML que je me suis mis à rechercher des informations sur le net. Bien vite une solution à été trouvée bien que pas simple d'utilisation. Elle consiste en l'utilisation de sections au sein du fichier de configuration. De cette manière, nous avons des éléments contenant plusieurs valeurs et non uniquement un couple « clé/valeur ».





Explication

*« Le principe est de créer un élément (la classe qui s'appelle ici ListeElement qui surcharge **ConfigurationElement**) qui seront contenus dans une collection d'éléments (ici la classe ListesElementCollection, qui hérite de **ConfigurationElementCollection**) et notre section qui va utiliser ces éléments (ici MesItems qui hérite de **ConfigurationSection**). »* (Nico-pyright, 2007)

Exemple du fichier de configuration

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>

<configSections>
<section name="MesItems"
type="Connection.Objets.Collection.MesItems, Connection" />
</configSections>

<!-- KRONOS -->
<appSettings>
<add key="ServeurOPC" value="OPC Kronos"/>
<add key="FrequenceAcquisition" value="20"/>
</appSettings>

<MesItems>
<mesitems nom="Couple charge 1" fonction="Vitesse véhicule"
typeValeur="decimal" valeur="0" id="1"/>
<mesitems nom="Vitesse véhicule" fonction="Vitesse véhicule"
typeValeur="decimal" valeur="0" id="2"/>
</MesItems>

</configuration>
```

Le dernier souci existant est le fait que le projet étant généré en fichier librairie dynamique, celui-ci ne possède pas de fichier *.config*, réservé a un exécutable. Le problème à été contourné en chargeant manuellement le fichier de configuration appelé « nomdeladll.dll.config ».

Exemple d'Utilisation

```
// Chargement du fichier Connection.dll.config (app.config de la
dll)
Configuration configConnect =
ConfigurationManager.OpenExeConfiguration("Connection.dll");

foreach (KeyValueConfigurationElement kvce in
configConnect.AppSettings.Settings)
{
	if (kvce.Key.Equals("ServeurOPC") == true)
		_serverProgID =
	configConnect.AppSettings.Settings[kvce.Key].Value;

	if (kvce.Key.Equals("ServeurHOST") == true)
		_serverProgHOST =
	configConnect.AppSettings.Settings[kvce.Key].Value;

	if (kvce.Key.Equals("FrequenceAcquisition") == true)
		Int32.TryParse(configConnect.AppSettings.Settings[kv
	ce.Key].Value, out _frequence);
}

// Capture de l'ensemble des Items défini dans le fichier
Connection.dll.config
MesItems section =
(MesItems)configConnect.GetSection("MesItems");

_ItemList.Clear();
_nbItem = 0;

_itemDefs = new OPCItemDef[section.Listes.Count];
_handlesSrv = new int[section.Listes.Count];

for (int i = 0; i < section.Listes.Count; i++)
{
	ListeElement element = section.Listes[i];
	AddItem(element.Nom, element.Fonction, element.TypeValeur,
	element.Valeur, element.Id);
}
```

6.3.2 Acquisition sur le serveur

La technologie OPC met en place un serveur qui met à disposition des objets appelés plus communément « Items » qui sont un ensemble de plusieurs propriétés dont notamment le nom de l'Item, sa valeur, son type de valeur.

Un client OPC s'abonne à un serveur OPC en s'y connectant et en y ajoutant un groupe. Ce groupe contient l'ensemble des Items dont la surveillance/manipulation est demandée par le client.

La connexion a un serveur OPC se fait au moyen de la librairie OPCdotNETLib qui fourni un ensemble de méthodes pour se connecter, ajouter des groupes, ajouter des items, effectuer des lectures/écritures et se déconnecter.

A chaque demande de lecture sur le serveur OPC via la librairie utilisée, celle-ci répondra en générant un évènement « ReadComplete » dont voici le détail du traitement :

```
/// <summary>
/// Event appellé lors d'une lecture sur le serveur OPC.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void _theGrp_ReadComplete(object sender,
ReadCompleteEventArgs e)
{
    try
    {
        foreach (OPCItemState s in e.sts)
        {
            List<Object> ItemListTemp = ItemList;

            foreach (Object obj in ItemListTemp)
            {
                Item<decimal> itDECIMAL = obj as Item<decimal>;

                if (itDECIMAL != null &&
s.HandleClient.Equals(itDECIMAL.id))
                {
                    itDECIMAL.valeur = (decimal)s.DataValue;
                }
            }
            ItemList = ItemListTemp;
        }
    }
    catch(Exception exe)
    {
            TraceLog(0,this.ToString(), exe.TargetSite.ToString(),
"Evènement de lecture impossible sur le groupe", e.ToString());
    }
}
```

Toutefois dans notre cas, il sera plus intéressant d'être averti par le serveur lors d'une modification d'une valeur afin de ne pas faire de lectures inutiles sur celui-ci. C'est donc d'une façon asynchrone que le serveur met à jour ses clients.

L'évènement reçu pour notifier cette mise à jour, c'est « _theGrp.DataChanged » qui sera attaché à un Handler identique à celui décrit ci-dessus.

La fréquence de mise à jour est définie par le troisième paramètre de la méthode de connexion au serveur :

```
// Ajout d'un groupe d'Items
_theGrp = _theSrv.AddGroup("OPCCSharp-Group", false, _frequence);
```

6.3.3 Mise à disposition des variables de la classe

Lors de chaque lecture sur le serveur ou mise à jour par celui-ci, l'évènement qui en découle modifie les variables d'une liste d'Items. C'est cette liste qui contient l'ensemble des Items spécifiés dans le fichier de configuration.

Une propriété existe pour modifier ou lire cette valeur. Celle-ci génère notamment un évènement quand la liste est modifiée. La classe « BACK.cs » n'a plus qu'à s'abonner à cet évènement pour recevoir la liste des valeurs modifiées.

Notons qu'un évènement similaire est déclenché pour prévenir du statut de la connexion au serveur.

6.4 Projet « Parsers »

Projet destiné à lire ou écrire des fichiers CSV.

Je n’utilise principalement qu’une méthode dans ce projet, celle consistant à lire un fichier CSV et retournant une List<List<Object>> comprenant une liste de lignes incluant elles mêmes une liste d’objets.

<u>Exemple de début fichier CSV</u>

```
nom dycle;Cycle europeen complet (20 minutes);;;;;;
tolerance en km/h;1;;;;;;
tolerance en sec;0,5;;;;;;
temps de prevention rapport;6;;;;;;
temps de passage rapport;6;;;;;;
;;;;;;;
;;;;;;;
;;;;;;;
Temps;Vitesse;debraye;Rapport;Consigne  Rapport;Acceleration;Pente;Distance
s;km/h;;;;m/(s*s);%;m
0;0;0;0;PM;0;0;0
1;0;;;;0;0;0
2;0;;;;0;0;0
3;0;;;;0;0;0
4;0;;;;0;0;0
5;0;;;;0;0;0
6;0;1;1;K1;0;0;0
7;0;;;;0;0;0
8;0;;;;0;0;0
```

6.5 Projet « LogAndException »

Ce projet à deux buts :

- Enregistrer les évènements dans un fichier
- Afficher des messages à l'utilisateur

6.5.1 Log d'évènements

La première action effectuée par l'objet une fois qu'il a été instancié via son constructeur par défaut est de supprimer les archives d'évènements remontant au delà d'une certaine date. Si le dossier n'existe pas, il est créé par défaut dans le dossier d'exécution du programme.

La deuxième action consiste en l'ouverture du fichier du jour s'il existe sinon il est créé sous la forme « log-YYMMDD.txt ». Ensuite, pour chaque instance du programme, une description de la machine est insérée dans le fichier en vue d'aider le développeur à comprendre la machine d'exécution. Ces renseignements sont de la forme suivante :

```
###############################################
# New instance of the program.

# Date                  : 5/05/2009
# Time                  : 16:09:02
# Machine               : HYENE
# User                  : daws
# Operating system      : Microsoft Windows NT 6.0.6001 Service Pack 1
# Processors            : 2
# Memory allowed(kb)    : 17576
# Framework             : 2.0.50727.3053
# Current directory     : C:\Users\daws\Documents\Visual Studio
2008\Projects\test\AutoCycle\bin\Release
###############################################
```





L'ensemble de ces renseignements sont obtenu via la variable « Environnement » du programme de la manière suivante :

```
ligne += "\n# Machine              : " + Environment.MachineName;
ligne += "\n# User                 : " + Environment.UserName;
ligne += "\n# Operating system     : " + Environment.OSVersion;
ligne += "\n# Processors           : " + Environment.ProcessorCount;
ligne += "\n# Memory allowed(kb)   : " + Environment.WorkingSet / 1024;
ligne += "\n# Framework            : " + Environment.Version;
ligne += "\n# Current directory    : " + Environment.CurrentDirectory;
```

Ensuite, en vue de remplacer un évènement, la fonction « `public void TraceLog(int level, string classe, string methode, string commentaire, string erreur)` » est utilisée pour ajouter l'évènement dans le fichier.

Notons que la fonction possède une propriété « level » qui sert à indiquer le niveau du message (INFORMATION, SUCCESS, ERROR) et ajoute les lignes d'erreur en fonction de ce niveau.

Exemple du fichier de log

```
Time        : 13:44:12
Statut      : INFORMATION
Classe      : Connection.Acquisitions.OPC.OPC
Méthode     : Start
Commentaire : Démarrage de l'acquisition
---------------------------------------------------------
Time        : 13:44:20
Statut      : ERROR
Classe      : Connection.Acquisitions.OPC.OPC
Méthode     : Void InitializeConfig()
Commentaire : 017: Problème avec la librairie de connection
Stack Trace : An XML comment cannot contain '--', and '-' cannot be the last
character. Line 8, position 5. (C:\Users\daws\Documents\Visual Studio
2008\Projects\test\AutoCycle\bin\Release\Connection.dll.config line 8)
---------------------------------------------------------
```

6.5.2 Messages utilisateur

Sert essentiellement à avertir l'utilisateur d'un évènement, d'une information ou encore d'une exception. Ensuite, ce même message est enregistré dans le fichier de log au moyen de la méthode vue plus haut.

Pour ce faire, l'objet désirant afficher un message doit appeler la fonction « `public void TraceMessageBox(int type, string classe, string methode, string number, string alternative, string erreur)` » en lui passant un numéro de message à afficher sous forme de texte pour le paramètre « number ». Ce message s'affiche alors à l'utilisateur au moyen d'une MessageBox.

Exemple de liste des messages

mb_msg_021	Please stop the cycle before resetting it.
mb_msg_020	Error during the file set up.
mb_msg_019	Template file doesn't exist.
mb_msg_018	End of the cycle.
mb_msg_017	Problem with the file connection. Please close the program, wait 30 seconds and restart the program.
mb_msg_016	Unable to disconnect from OPC server.
mb_msg_015	Unable to read Items on the OPC server.
mb_msg_014	Unable to activate Items group.
mb_msg_013	Unable to add the server handles for the items.
mb_msg_012	Items have not been valited by the OPC server.
mb_msg_011	Unable to add Items to Items group.
mb_msg_010	Unable to add Items group to OPC server.
mb_msg_009	Unable to connect to OPC server.
mb_msg_008	Unable to communicate with OPC server
mb_msg_007	The opening of the options box during a cycle may create incoherences at the axe auto-adjustement level. Do you want to continue ?
mb_msg_006	No cycle have been done.
mb_msg_005	Load a cycle before resetting it.
mb_msg_004	Start the communication with OPC Server before running cycle.
mb_msg_003	Load a cycle before running it.
mb_msg_002	Unable to load another cycle when an existing cycle is running.
mb_msg_001	Communication is already started.

6.5.3 Utilisation de ces fonctions

Pour pouvoir utiliser ces fonctions, chaque objet doit avoir la référence de l'objet « LogAndException » dans ses variables membres. Cette référence est passée lors de l'instanciation par le constructeur et ce, lors du lancement du programme par la classe « BACK.cs ».

Dans le cas ou l'objet « LogAndException » n'as pas pu être créé et donc que la référence est nulle, les fonctions peuvent êtres utilisées mais ne sont pas redirigées vers l'objet de log afin d'éviter tout plantage. Par conséquent, sans cet objet, pas de log ni de message utilisateur.

6.6 Projet « UC_Statut_button »

Contrôle utilisateur destiné à afficher le statut de l'embrayage sur l'interface principale. Cet élément n'est pas dans le cahier des charges mais me semble non pas indispensable non plus mais bien une aide non négligeable pour l'opérateur.

Quand celui-ci est, par exemple, dans une zone du cycle ou il doit avoir mis le premier rapport de transmission mais que l'embrayage est enfoncé et donc que le rapport n'est pas enclenché, il est important de l'en informer.

Cette image peut donc contenir deux types d'information :

- « RELACHER » (l'embrayage) : image verte avec une écriture noire.
- « POUSSER » (l'embrayage) : image rouge avec une écriture jaune.

Le choix des couleurs est fait pour que l'image soit le plus visible par l'opérateur et en se basant sur le fait que un actuateur ne nécessitant pas d'opération doit être, comme un capteur, sur l'état « OK » ou plus généralement de couleur verte.

De plus, cette image contient une bordure pour renfoncer le contraste et est disponible dans toutes les langues proposées à l'opérateur.



Figure 6-8 Embrayage - vert – français



Figure 6-9 Embrayage - rouge - français



Figure 6-10 Embrayage - vert - anglais

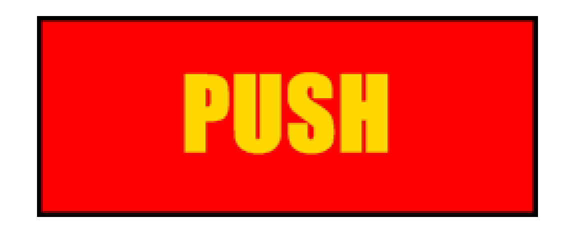


Figure 6-11 Embrayage - rouge - anglais

6.7 Projet « UC_Blink_fleche »

Projet non demandé dans le cahier des charges mais, encore une fois, qui se révèle d'une aide certaine pour l'opérateur.

Le concept de ce projet est d'avertir ce même opérateur d'un changement de rapport futur. Ce module peut être considéré comme un module de prévention de changement de rapport de vitesse.

Il vient se placer dans la colonne de droite avec les autres affichages et est divisé en deux parties distinctes mais fonctionnant simultanément. Une flèche pour indiquer si le prochain rapport est supérieur ou inférieur au rapport actuel et, un décompte en seconde pour afficher le temps restant avant le changement.

Lors de ce décompte, la flèche clignote à une fréquence de 250ms pour attirer l'œil de l'opérateur. Une fois le décompte terminé, le contrôle d'utilisateur se fige avec un temps à 0 seconde, la flèche affichée dans le bon sens et ce, pendant un certain laps de temps (en secondes) ou le fond du contrôle sera coloré différemment.

Dépendances spécifiques

- Owf.Controls.DigitalDisplayControl

6.7.1 Configurations

Le temps de décompte et de fixation du contrôle en une couleur différente est défini dans le fichier de cycle. En effet, un cycle peut avoir un palier de changement de rapport de transmission plus long ou moins long que celui utilisé par défaut. (Le cycle NEDC)

De même, le temps de fixation est laissé à l'appréciation de l'opérateur.

Fichier de cycle

nom du cycle	Cycle européen complet (20 minutes)
tolérance en km/h	1
tolérance en sec	0,5
temps de prévention rapport	6
temps de passage rapport	6

La couleur de fixation est quand à elle configurée via le panneau d'outils.

6.7.2 Problèmes rencontrés

6.7.2.1 Affichage d'un chiffre

Le contrôle utilisateur à été conçu pour n'afficher qu'un seul chiffre lors du décompte du temps. Par ce fait, tout nombre supérieur à 9 (et donc comportant au minimum deux chiffres) sera changé par le chiffre 9 qui est donc par conséquent le décompte maximal autorisé.

6.7.2.2 Fixation de l'interface utilisateur

Au tout début du développement, aucun thread n'était mis en place pour ce contrôle utilisateur ce qui fait que l'interface générale était figée lors des « Thread.Sleep(secondes) ; ».

Lors de chaque prévention de rapport, un thread est donc créé pour s'occuper du changement de flèche, du décompte ainsi que des fixations diverses.

6.7.2.3 Décompte non synchronisé

Lors de l'exécution du contrôle utilisateur, il s'est avéré qu'il existait un décalage temporel entre le passage du rapport et la fin de la fixation du contrôle.

Utilisant un timer, celui-ci effectue son premier tick après le laps de temps prédéfini. Or, le programme n'exécutait le décompte qu'à partir du premier tick.

Donc, si le décompte commence à 3 secondes et que le tick est de 250 ms, celui-ci terminera 250 ms trop tard car en réalité, ce laps de temps de 250 ms à déjà été écoulé. Le temps restant après le premier tick est donc de 2,75 secondes et non 3 secondes. Le problème à été résolu à l'aide d'une simple soustraction.

6.7.3 Exemple d'utilisation

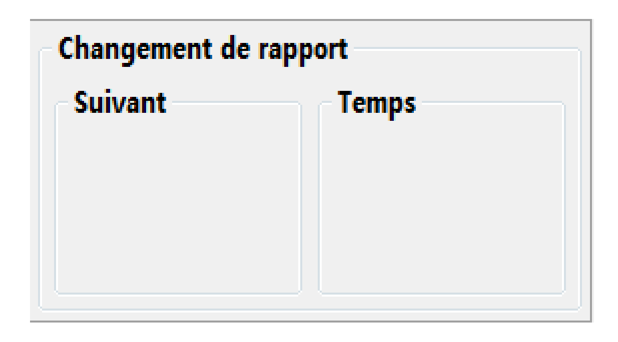


Figure 6-12 Prévention de rapport - Avant

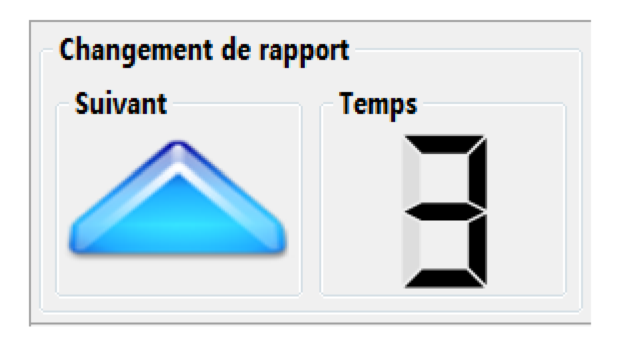


Figure 6-13 Prévention de rapport - Pendant (3 secondes restantes, rapport montant)



Figure 6-14 Prévention de rapport - Pendant (2 secondes restantes)



Figure 6-15 Prévention de rapport - Zone de changement de rapport

6.8 Projet « Rapport »

En vue de réaliser l'archivage d'un cycle de conduite réalisé, il est important d'en créer un rapport. Celui-ci contient l'ensemble des valeurs collectées par le programme et inclura quelques valeurs supplémentaires.

6.8.1 Type de fichier

En vue de générer un rapport, la question du type de fichier de sortie s'est imposée. Devais-je sortir le graphique sous forme d'image, générer le rapport en format PDF, Word, … ou bien d'autres possibilités.

Restreint par le fait de garder le maximum de donnée et de pouvoir manipuler celles-ci, le choix de l'utilisation d'un tableur est devenu inévitable.

Sachant que la majorité de l'administration et des utilisateurs utilisent la suite Microsoft Office, c'est donc avec le logiciel Microsoft Excel que j'ai commencé à me documenter à la place de rechercher des informations sur d'autres formats (type OpenOffice.org).

Bien vite, je me suis tourné vers le format .XSLX utilisant la norme Office Open XML. Les raisons de ce choix sont expliquées au paragraphe 5.2.8.

6.8.2 Choix des résultats de sortie

- ***Feuille récapitulative***
  - Date
  - T° ambiante
  - P° ambiante
  - Nombre de points mesurés
  - Nombre de points sortis des valeurs
  - % de points hors tolérance
  - Erreur relative moyenne en %

- ***Feuille pour insérer des graphiques***

- ***Feuille des mesures réalisées***
  - Temps (secondes)
  - Vitesse théorique (km/h)
  - Vitesse réelle (km/h)
  - Lambda
  - Débit d'air
  - Consommation (g/s)

6.8.3 Fonctionnement

Lors de chaque rafraichissement de données, celles-ci sont mises dans une liste comprenant pour chaque temps de mesure, la totalité des mesures.

Lors de la génération de rapport, il suffit alors de réinsérer ces mesures dans les feuilles du fichier de rapport.

Pour générer le fichier, je me sers d'un modèle comprenant le nom des colonnes et des divers champs avec les feuilles déjà existantes. Le programme en crée une copie sous un autre nom et insère les valeurs dans les cellules respectives.

Pour calculer le pourcentage d'erreur relative moyenne, on divise la moyenne de la vitesse théorique sur la moyenne des écarts. Le tout multiplié par 100 pour avoir un pourcentage.

6.8.4 Problèmes rencontrés

6.8.4.1 Nombre de points

Le nombre de points étant trop important (dépassant les centaines de milliers selon le cycle réalisé et la fréquence d'acquisition) pour Microsoft Excel, les résultats ne sont plus représentés que chaque seconde dans le rapport. Ainsi, le programme va préalablement calculer une moyenne pour chaque seconde avant de l'insérer dans le rapport.

De ce fait, la vitesse de génération du rapport est passée de quelques minutes à quelques secondes. Autre initiative pour améliorer la vitesse de traitement, les points météos ne sont inscrit que une seule fois dans le rapport et ce, sur la feuille « Main » regroupant les données principales du cycle.

6.8.4.2 Bugs

La librairie « ExcelPackage » étant relativement récente, elle comporte encore quelques bugs qui m'ont fait perdre de nombreuses heures.

Le premier étant le fait que lorsque plusieurs feuilles sont ouvertes dans le document, il est OBLIGATOIRE de modifier au moins une cellule de chaque feuille avant de sauver le document, sinon celui-ci génère une exception.

Celui-ci à été résolu en recopiant le contenu de chaque première cellule de chaque page par son propre contenu. Inutile mais obligatoire pour contrer le bug.

Exemple

```
// Update d'une cellule de chaque worksheets pour éviter
// le bug lors de la fermeture du fichier lorsqu'il
// existe plusieurs worksheets
worksheetMain.Cell(1, 1).Value = worksheetMain.Cell(1, 1).Value;
worksheetGraph.Cell(1, 1).Value = worksheetGraph.Cell(1, 1).Value;
worksheetDataMoy.Cell(1, 1).Value = worksheetDataMoy.Cell(1, 1).Value;
```

Le deuxième bug concerne l'utilisation de formules dans le fichier. Excel ne recalcule pas les formules du fichier lors de l'ouverture. Dans mon cas, il existe une formule de pourcentage pour le nombre de points hors valeurs qui est prédéfinie dans le fichier servant de modèle.

Dans ce fichier de modèle, la formule renvoie bien entendu une erreur (DIV/0) car aucune cellule pour la formule n'est remplie. Mais, une fois ces données remplies dans le rapport, Excel n'en recalcule pas la formule. Il faut donc sélectionner la cellule et forcer la mise à jour. Une solution est expliquée sur le site de la librairie mais celle-ci génère systématiquement une exception.

6.8.5 Améliorations ajoutées

6.8.5.1 Nom du fichier

En vue de faciliter le classement des fichiers de rapport, le nom de fichier proposé à l'utilisateur est sous la forme « YYMMJJ-hhHmmMssS-nomdufichiervoulu.xlsx ».

6.8.5.2 Fenêtre de visualisation

Elle permet de voir l'état d'avancement du calcul des moyennes ainsi que de l'insertion dans les fichiers. L'opérateur peut donc se rassurer sur le temps nécessaire pour la création du rapport.

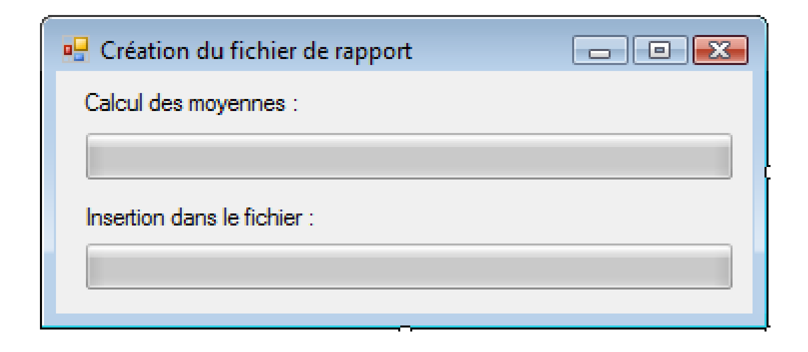


Figure 6-16 AutoCycle - Génération de rapport

6.9 Techniques spécifiques utilisées

6.9.1 Appels inter-threads

Depuis l'apparition du Framework .NET 2.0, un thread autre que le thread parent d'un contrôle ne peut plus accéder directement à une méthode ou une propriété de ce contrôle car cela mène à des résultats imprévisibles.

De ce fait, il existe (toujours depuis le Framework .NET 2.0) une nouvelle propriété appelée « CheckForIllegalCrossThreadCalls » prenant comme valeur un booléen et permettant de désactiver la vérification des appels inter-threads. Cette propriété, si elle est activée, émet une exception lors d'un appel inter-thread non sécurisé.

Notons que de manière plus générale, cette propriété existe en vue de méthodes de débogage mais qu'elle peut être utilisée si l'on ne veut pas utiliser des opérations dites « thread-safe ».

Dans mon cas, lors du début de la conception du programme et même pendant l'entièreté du premier mois, j'utilisais cette propriété de manière désactivée afin de simplifier le travail. Après pas mal de tests, il s'est avéré que cela ne changeais strictement rien. Et pour preuve, la majorité des applications développées avant ce Framework n'effectuait pas d'appels inter-threads de manière sécurisée.

Mais bon, sachant que l'on désire coder le plus proprement possible, je me suis plié à la volonté de la documentation MSDN qui explique que cette propriété doit être utilisée uniquement pour le débogage.

Pour ce faire, il existe plusieurs méthodes telles que l'utilisation de *delegate* et *events* de manière synchrone ou asynchrone.

De manière synchrone (celle utilisée dans notre exemple), aucune valeur de retour n'est attendue et l'appel attend que la méthode soit finie avant de continuer son exécution. C'est donc un mode bloquant destiné aux instructions d'une durée relativement faible.

De manière asynchrone, la méthode est appelée et exécutée alors que l'appel de la méthode rend directement la main. C'est donc un mode non bloquant. Plus pratique, mais plus compliqué à mettre en place.

Pour la suite, je ne vais expliquer que la méthode synchrone. Celle-ci utilise la méthode « Control.Invoke(delegate, object[]) » ou « Control.Invoke(delegate) » quand aucun paramètre n'est requis.

Exemple de code

```
#region Méthodes pour les operations THREAD-SAFE
/// <summary>
/// Permet d'afficher la PictureBox contenant la flèche
/// </summary>
private void arrowShow()
{
    // InvokeRequired required compares the thread ID of the
    // calling thread to the thread ID of the creating thread.
    // If these threads are different, it returns true.
    if (this.PB_Arrow.InvokeRequired)
    {
        arrowShowCallback d = new arrowShowCallback(arrowShow);
        this.Invoke(d);
    }
    else
    {
        this.PB_Arrow.Show();
    }
}
```

this.PB_Arrow.InvokeRequired

Va comparer le numéro d'identification du thread qui appelle la méthode et celui du thread propriétaire de l'objet. Si celui-ci est différent il rentre dans la fonction.

arrowShowCallback d = new arrowShowCallback(arrowShow)

Création d'un nouveau délégué comprenant la méthode « arrowShow ».

this.Invoke(d);

Exécute le délégué spécifié sur le thread qui possède le contrôle.

Remarque : Un délégué est une structure de donnée qui fait référence à une méthode statique ou à une instance de classe et une instance d'une méthode de cette classe.

6.9.2 RessourceManager

Le ResourceManager est un manageur de fichiers de ressources. Il permet de charger un fichier de ressources et d'en extraire les valeurs.

Ce fichier a diverses utilités et, dans notre premier cas, est utilisé pour garder en mémoire tout les paramètres définis par l'utilisateur. Il fonctionne sous forme de clé/valeur/commentaire et ce sont des valeurs par défaut.

Toute notre boite d'outils et les choix de configurations faits par l'opérateur sont enregistrés dans ce fichier pour la prochaine instance du programme. De même que si une erreur à été commise, il est possible de restaurer les valeurs par défaut de ce fichier.

6.9.3 ApplicationBinding

L'ApplicationBinding est l'équivalent d'un DataBinding mais au niveau de l'application. Je m'en sers pour binder des propriétés des ressources de l'application à notre programme et ce, d'un point de vue application.

Par exemple je change la couleur d'un objet, ce changement va se répercuter au niveau des propriétés de l'application et cette même application va prévenir tout les membres abonnés à cette propriété.

Les propriétés sont donc liées aux valeurs du fichier de ressource, c'est pourquoi nous avons un fichier de ressource principal contenant l'ensemble des valeurs utilisées tels que la langue, les couleurs, les délais, … .

6.9.4 Localisation

L'application est conçue pour être multilingues. Chaque fenêtre possède son fichier de ressources pour modifier les noms des labels etc. De cette manière, chaque instanciation de fenêtre va chercher le fichier de ressources adapté à la langue actuelle du programme (la culture).

Afin de pouvoir dynamiquement changer la langue sans relancer le programme, j'utilise une propriété du fichier ressource liée à l'ApplicationBinding. Chaque fenêtre peut alors recevoir le changement de langue, charger le bon fichier et modifier les valeurs de ses membres.

6.9.5 Diagnostique

Lors de l'exécution du programme, il s'avère que l'occupation du processeur est anormalement élevée pour notre programme. La question est donc de savoir si les centaines de calculs effectués à une certaine période en sont la cause ou si la faute est à reporter sur le graphique.

Plus étonnant, après quelques tests à une fréquence de rafraichissement de 20 ms il s'est avéré que la fréquence réelle était de 50 ms. L'occupation du processeur déteint donc sur les performances du programme.

C'est donc avec le but de comprendre quelle partie est en cause que j'ai cherché des outils de diagnostique. La classe « Stopwatch », qui existe depuis le Framework .NET 2.0 à été utilisée.

Celle-ci permet des mesures précises de temps écoulés, contrairement à d'autres classes ou à l'utilisation de l'heure.

Il s'est donc avéré que la partie calculs du programme ne consomme même pas 1 ms quand dès lors la partie graphique en consomme davantage. La raison est expliquée plus haut dans le choix de la librairie graphique, au point 5.2.5.

6.9.6 Déploiement

La solution possède un projet de déploiement offert par Microsoft Visual Studio 2008 qui permet de définir de multiples paramètres pour créer un fichier d'installation. Il est notamment possible de définir le dossier par défaut, ajouter des raccourcis dans n'importe quel endroit (sur le bureau, dans le menu démarrer, …) et bien d'autres possibilités.

Le fichier d'installation fourni est du type « Setup.exe » ou tout autre nom voulu.

7 Tests de l'application

7.1 Ordinateur livré avec le banc

Caractéristiques

- **Processeur :**
- **Mémoire ram : 256 mo SD-RAM**
- **Disque dur :**
- **Carte graphique :**

7.1.1 But

Cette série de tests consiste à vérifier la cohérence du projet avec l'ensemble du matériel disponible. Plus concrètement :

- Vérifier qu'il est possible d'utiliser l'entièreté du matériel (capteurs, communication, OBD, …) avec le logiciel AutoCycle.
- Vérifier que la machine peut faire tourner notre application dans des conditions optimales.

7.1.2 Tests réalisés

7.1.2.1 Communication

J'ai d'abord réalisé l'acquisition sur le serveur OPC au moyen d'un logiciel déjà existant. Ainsi, j'ai pu comprendre le fonctionnement de la communication OPC et voir le type de variables retournées par le serveur.

Toutefois, il s'est avéré impossible d'effectuer une connexion au serveur OPC via le réseau. Depuis Microsoft Windows XP Service Pack 2, les règles de sécurités ont été mise à jour et ne laissent plus passer les objets de type DCOM qui sont utilisés pour la communication OPC.

Après de vaines tentatives de modification du firewall et un échange avec le professeur de l'école responsable du module OPC de l'école, cette partie me semblait trop hasardeuse donc je l'ai simplement abandonné. Néanmoins cette communication reste théoriquement possible pour le futur.

7.1.2.2 Fréquence d'acquisition sur le boitier d'acquisition

Par défaut, Kronos possède une fréquence de rafraichissement de 240 ms sur le boitier d'acquisition. Il est apparu bien vite que cette fréquence était beaucoup trop faible.

L'opérateur devant rester dans un intervalle de deux km/h par rapport à la vitesse théorique ne pouvait pas respecter cette contrainte et ce, particulièrement lors des phases d'accélération et de décélération.

En effet, l'opérateur voit sa vitesse se rafraichir plus ou moins quatre fois par seconde alors que pendant ce temps, la vitesse peut avoir changé soixante fois. Il n'a donc la possibilité de corriger sa vitesse que quatre fois par seconde. Cela consiste en un effet d'escalier visible sur le graphique. Cet effet est représenté à la page suivante.

Le but ici est donc d'augmenter la fréquence d'acquisition du serveur afin de faciliter au maximum la correction de la vitesse par l'opérateur.

L'idéal est d'obtenir un taux de rafraichissement de 24 images par seconde ce qui correspond approximativement à la perception de l'œil humain. (Bien qu'en pratique, le fonctionnement de l'œil humain ne se base pas sur un taux de rafraichissement mais ceci est un autre sujet)

Après une série de tests, il s'est avéré que le serveur commence à planter à une fréquence supérieure à 50 Hz. L'objectif à donc été placé à une fréquence de 50 Hz.

En résumé, l'application Kronos rafraichi ses données toutes les 20 ms et l'application AutoCycle est averti des changements de données à une fréquence X de mise à jour par le serveur OPC.

Cette valeur X doit être égale a la fréquence d'actualisation des capteurs par Kronos car dans le cas ou elle est plus faible il y a perte de données et dans le cas ou elle est plus élevée, il y a des calculs inutiles vu que les données n'ont pas eu le temps de changer.

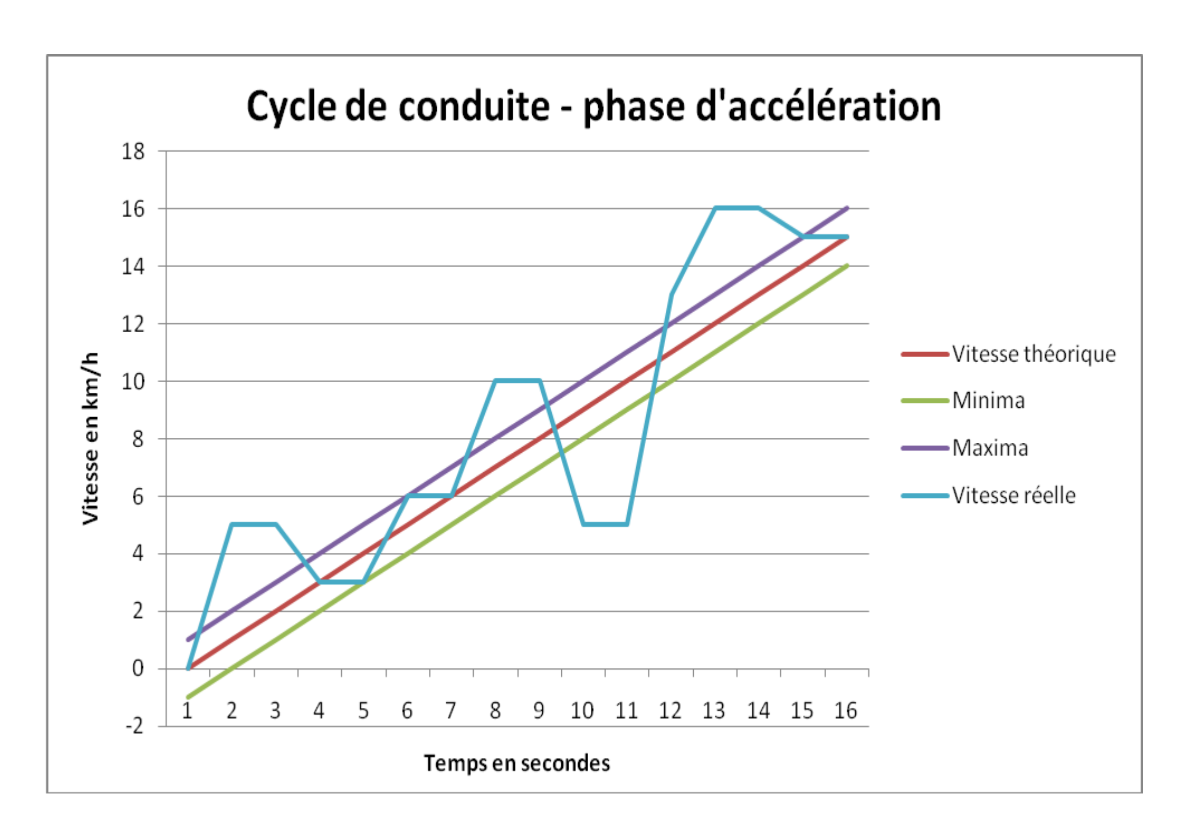


Figure 7-1 Cycle avec faible fréquence d'acquisition

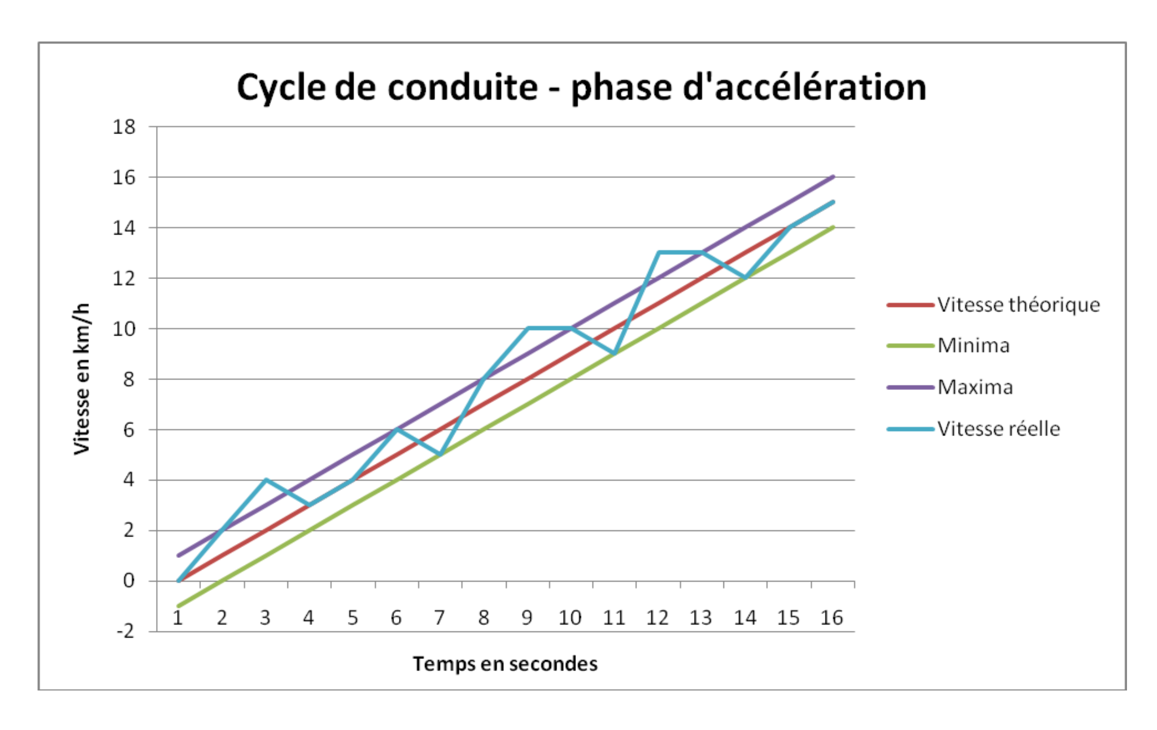


Figure 7-2 Cycle avec haute fréquence d'acquisition

7.1.2.3 Fréquence d'acquisition du graphique

Il faut savoir que le logiciel AutoCycle possède deux types de rafraichissement des données. Le premier consiste en la fréquence de mise à jour par le serveur OPC des données des capteurs alors que le deuxième concerne le rafraichissement du graphique pour l'opérateur.

Concernant la première fréquence, elle est déjà expliquée au point précédent et, pour rappel, possède un taux de rafraichissement de 50 Hz.

La fréquence propre au graphique consiste à déterminer le nombre de fois par seconde que le graphique va prendre les données des capteurs et les ajouter sur le graphique. (Et bien entendu faire tout les calculs de centrage de celui-ci)

Cette fréquence doit donc être identique à la fréquence de mise à jour par le serveur OPC et ce, pour les mêmes raisons évoquées au précédent point.

Dans le cas ou cette fréquence n'arrive pas à être supportée cela amène à un effet d'escalier ; problème identique évoqué au précédent point.

Après une série de tests réalisés, il est apparu que l'application n'arrivait pas à soutenir une fréquence d'acquisition élevée et ce, en raison de la charge du processeur qui saturait. Cette charge est créée par le rafraichissement du graphique.

N'ayant aucune autre solution que de changer le processeur, il a été décidé de remplacer l'ordinateur du banc par un plus puissant. J'ai donc, pour juger le matériel adapté, réalisé des tests sur une machine temporaire. Ceux-ci sont disponibles au point suivant.

7.2 Ordinateur personnel

Caractéristiques

- **Processeur :** *Intel T7300 - 2.0 GHz – 4Mo cache*
- **Mémoire ram :** *2 Go DDR2*
- **Disque dur :** *120 Go 5400 tr/min*
- **Carte graphique :** *NVidia GeForce 8400M*

7.2.1 Test 1

Date locale	24/03/2009 16:29:50
Fréquence de rafraichissement en Hz	50
T° ambiante en °C	17,80000114
P. ambiante	0,967100037
Nombre de points mesurés	19934
Nombre de points sortis des valeurs	2645
% de points hors tolérance	13,268787
Erreur relative moyenne en %	0,412043135
durée du test en secondes	620
moyenne de points par seconde	32,1516129

Test réalisé avec la voiture sur le banc et représentant la partie un du cycle de conduite NEDC. On peut apercevoir que l'on obtient une bonne moyenne de points par seconde. Cependant, la charge du processeur reste assez élevée mais ne sature pas.

7.2.2 Test 2

Date locale	10/04/2009 13:38:09
Fréquence de rafraichissement en Hz	50
T° ambiante en °C	19,5
P. ambiante	0,964799988
Nombre de points mesurés	3189
Nombre de points sortis des valeurs	1637
% de points hors tolérance	51,33270618
Erreur relative moyenne en %	21,28318758
durée du test en secondes	106
moyenne de points par seconde	30,08490566

Test réalisé avec un simulateur de vitesse et servant à vérifier la moyenne de points par seconde. On peut constater que la moyenne est légèrement inférieure au test précédent mais que celle-ci reste suffisante.

7.2.3 Test 3

Date locale	10/04/2009 13:42:04
Fréquence de rafraichissement en Hz	50
T° ambiante en °C	19,5
P. ambiante	0,964700012
Nombre de points mesurés	929
Nombre de points sortis des valeurs	113
% de points hors tolérance	12,16361679
Erreur relative moyenne en %	1,981134896
durée du test en secondes	27
moyenne de points par seconde	34,40740741

Test réalisé avec un simulateur de vitesse et servant à vérifier la moyenne de points par seconde. On peut constater que la moyenne est nettement supérieure au test précédent alors que le test à une durée beaucoup plus courte.

7.2.4 Test 4

Date locale	10/04/2009 14:03:38
Fréquence de rafraichissement en Hz	50
T° ambiante en °C	19,5
P. ambiante	0,964700012
Nombre de points mesurés	30911
Nombre de points sortis des valeurs	4395
% de points hors tolérance	14,21823946
Erreur relative moyenne en %	0,695040625
durée du test en secondes	1219
moyenne de points par seconde	25,35767022

Test réalisé avec la voiture sur le banc et représentant l'entièreté du cycle de conduite NEDC. On peut apercevoir que l'on obtient bonne moyenne de points par seconde vraiment inférieure aux précédents tests, à la limite du tolérable (24 images / seconde).

Quand au processeur, il a augmenté sa charge d'environ 15% et la consommation de mémoire vie par le processus est passée de 30 à 40 mo.

7.2.5 Conclusion

Je peux donc en déduire que la machine de test employée s'avère suffisante mais pas optimale car il y a quand même un risque de saturation.

De plus, ces tests ayant été effectués sous Microsoft Windows Vista qui intègre un grand nombre de processus fonctionnant en arrière tache de manière variable.

Il n'est donc pas possible de comparer ces tests de manière optimale à cause de la charge du processeur changeant indépendamment du logiciel AutoCycle.

J'en conclu donc que les fréquences d'acquisition sont mieux respectées avec une machine plus puissante. Il est donc nécessaire de remplacer la machine actuelle.

7.3 Nouvel ordinateur

Caractéristiques

- **Processeur :** *Intel Q9550 – 2,83 GHz – 12 Mo cache*
- **Mémoire ram :** *4 Go – DDR3*
- **Disque dur :** *80 Go SATA 2 7200tr/min*
- **Carte graphique :** *NVidia GeForce 9500GT PCI-E*

7.3.1 But

Cette séquence de test prend en compte l'utilisation du nouvel ordinateur, qui a remplacé l'ancien. Il est donc important de réaliser un test comparatif pour voir si le changement de machine à répondu à nos attentes ainsi que d'éventuellement constater des différences.

7.3.2 Test réalisé

Date locale	26/05/2009 17:31:43
Fréquence de rafraichissement en Hz	50
T° ambiante en °C	21,4
P. ambiante	0,972
Nombre de points mesurés	39071
Nombre de points sortis des valeurs	4933
% de points hors tolérance	12,6257
Erreur relative moyenne en %	0,52824
durée du test en secondes	1219
moyenne de points par seconde	32,0516129

7.3.3 Interprétations

1. Comparé au test réalisé au point 7.2.4 (réalisé sur une machine différente mais avec les mêmes paramètres) qui lui comptait 30911 mesures, nous en obtenons ici 39071. C'est donc une amélioration de 8160 mesures ou de 26%, le tout par rapport à l'ancien test.
2. Non observable dans le tableau ci-dessous, la moyenne de point par seconde est restée identique du début à la fin du test (+/- 32 mesures/seconde). Or, dans les tests effectués sur d'autres machines, la moyenne par seconde est élevée au début du test et faible à la fin de celui-ci.

7.3.4 Conclusions

Tout d’abord, et en toute logique vu la puissance de calcul de la machine, le programme consomme environ 25% du processeur, ce qui est nettement plus faible qu’avant et permet donc de ne plus saturer la machine. Il est donc possible, grâce à cette non-dégradation de performances, de conserver une moyenne stable de mesures par seconde.

Ensuite, vu la marge d’inactivité laissée par le processeur, il est possible d’augmenter la fréquence de rafraichissement de l’acquisition et de l’affichage. Cela permet donc une meilleure facilité visuelle pour l’opérateur.

L’objectif de ce test, à savoir prouver l’efficacité de la nouvelle machine, est réussi.

8 Conclusions

Tout d'abord, ce projet était un défi personnel car j'ai délibérément choisi ce stage sachant pertinemment bien qu'il n'y aurait pas d'informaticien sur place pour me guider lors d'éventuels problèmes. J'ai finalement réussi à gérer l'ensemble du projet seul en y apportant des améliorations, suggestions et en faisant appel à ma débrouillardise personnelle pour trouver une solution aux problèmes rencontrés.

Je suis personnellement satisfait du résultat du projet et donc, de la réussite de mon propre défi.

Concernant la méthode journalière de travail, j'ai pu essayer un mode de travail de plus en plus couru dans le monde informatique, celui du télétravail.

Effectivement, n'ayant pas besoin de me déplacer tout les jours au campus, j'ai effectué une partie de mon travail à domicile.

J'en retire donc la grande souplesse horaire offerte par le télétravail.

D'un point de vue technique, pour la technologie OPC, j'ai rencontré énormément de difficultés à plusieurs niveaux. J'ai finalement pu résoudre ceux qui étaient possible d'être résolus par moi-même.

J'en conclus donc pour cette technologie que celle-ci à tout pour plaire en théorie mais que en pratique, celle-ci est très compliquée à mettre en place et n'est pas fiable. En effet, à une très haute fréquence d'acquisition, le serveur OPC présentait quelques instabilités avec pertes d'informations.

Concernant le projet, il a été abouti dans les temps et le cahier des charges à été respecté dans son entièreté. La plus grosse partie du temps à été prise par la recherche de solutions aux très nombreuses contraintes rencontrées, et non au développement du code.

Le projet fini rempli donc son objectif qui était de développer un logiciel de gestion de cycles de conduite pour le campus et son banc à rouleaux car il n'existait aucune offre logicielle de ce type.

De plus, le logiciel, pouvant prendre en compte n'importe quel type de cycle, permet de réaliser un grand nombre d'études et non pas seulement l'étude d'émission de pollution comme fixée au début ; ce qui est une plus value non négligeable au projet.

Pour le futur, il est envisagé de soit continuer le développement de ce logiciel soit de développer un logiciel complémentaire en vue de réaliser d'autres usages spécifiques comme des tests sur les consommations, par exemple.

Ce projet est donc une bonne base complète et fonctionnelle qui va servir pleinement afin de réaliser des tests sur le banc à rouleaux et peut-être, au futur, à d'autres usages.





9 Glossaire

- **API** : Application Programming Interface. Interface servant de passerelle entre le programme et le système d’exploitation.
- **C# :** langage de programmation orienté objet de Microsoft.
- **Capteur :** dispositif transformant une grandeur physique en une grandeur utilisable.
- **CPU :** Central Processing Unit. Processeur central de l’ordinateur.
- **CSV :** Comma-separated values. Format informatique.
- **DCOM :** Distributed Component Object Model. Créé par Microsoft, cette technologie est utilisée pour le développement objet et définit les spécifications pour la distribution et l’accès sur le réseau d’applications.
- **Dll :** Dynamic Link Library. Bibliothèque de classes, fonctions qui peut être chargée de manière dynamique.
- **Ethernet :** protocole de communication réseau de bas niveau.
- **Firewall :** logiciel/matériel qui à pour but de filtrer les entrées/sorties du réseau.
- **GPU :** Graphics Processing Unit. Processeur de la carte graphique.
- **HEPL :** Haute Ecole de la Province de Liège.
- **Hz :** hertz. Unité de mesure périodique représentant un nombre d’actions par seconde.
- **JAVA :** langage de programmation orienté objet de Sun Microsystems.
- **MSDN :** Microsoft Developer Network
- **NEDC :** New European Driving Cycle. Cycle de conduite européen utilisé pour calculer la pollution d’un véhicule.
- **Norme :** directives qui régissent un domaine. Standard.
- **OBD :** On Board Diagnostics. Système de surveillance des éléments du véhicule.
- **Octet :** unité de mesure comprenant 8 bits. (utilisé pour définir une quantité d’informations)
- **OLE :** Object Linking and Embedding. Fonctionnalité permettant le partage de données au sein de documents indépendament du logiciel.
- **OPC :** OLE for Process Control. Apparue en 1995, cette technologie est dédiée à l’interopérabilité des systèmes industriels. Elle est basée sur la communication réseau DCOM.
- **Protocole :** ensemble de règles et normes définissant les échanges entre les ordinateurs.
- **Protocole propriétaire :** protocole dont les règles et normes ne sont pas publiques.
- **TCP/IP :** protocole de communication réseau basé sur le mode connecté et l’utilisation d’adresses IP (Internet Protocol).
- **Thermocouple :** système de mesure de température.
- **Thread :** processus léger.
- **Tick :** évènement déclenché par la minuterie. (Exemple : 1 tick toute les secondes)
- **Timer :** minuterie.
- **XML :** Extensible Markup Language. Standard ouvert utilisant des balises pour décrire des données.

10 Licences

10.1 Licences utilisées

- **GPL :** (Anglais : General Public License). Licence selon laquelle un logiciel doit être diffusé avec ses sources et peut être librement adapté et modifié. La licence est dite contaminante, puisque tout logiciel désireux d'utiliser tout ou partie d'un logiciel GPL existant tombe lui-même automatiquement sous le coup de la GPL.

- **LGPL :** Licence publique générale limitée GNU. Version allégée de la GPL qui permet à un programmeur de sortir un programme propriétaire lié à un programme open-source sous licence LGPL sans en devoir livrer le code source.

10.2 Licences respectives

- **Librairie Zedgraph :** *LGPL.*
- **Librairie OPCdotNETLib :** *pas de licence.*
- **Librairie ExcelPackage :** *GPL.*
- **Librairie Owf.Controls :** *pas de licence.*
- **Microsoft Visual Studio 2008 version étudiante :** *les logiciels sont destinés à un usage privé et non commercial.*

10.3 Licence d’AutoCycle

Mon logiciel utilisant des librairies ayant des licences de type GPL et LPGL, c’est la licence la plus contraignante qui domine et donc dans ce cas la GPL.

De plus, développé avec le logiciel Microsoft Visual Studio 2008, le logiciel ne pourra pas être utilisé de manière commerciale.

Si la volonté est d’en faire un usage commercial il en convient donc d’utiliser un autre environnement de développement ou la version gratuite de Microsoft Visual Studio (version express). De plus, la librairie ExcelPackage ne pourra plus être utilisée mais sachant que le format .XLSX est ouvert, il suffira au programmeur de recoder cette librairie. Ainsi, la licence, utilisant que des librairies de licence LPGL, pourra être utilisée à des fins commerciales.

Actuellement, le logiciel AutoCycle V1.0 est donc sous licence **GPL**.

11 Bibliographie

- Campus Automobile de Spa-Francorchamps. (s.d.). *Dossier de presse.* Consulté le Mai 12, 2009, sur http://cms.horus.be/files/99920c/MediaArchive/Presse/Dossier_Presse_Campus_10sept08.pdf
- Communautés Européennes. (2004). *DIRECTIVE DU CONSEIL concernant le rapprochement des législations des États membres relatives aux mesures à prendrecontre la pollution de l'air par les émissions des véhicules à moteur.* Norme, Office des publications officielles des Communautés européennes.
- *Explaining Delegates in C# - Part 3 (Events again)*. (s.d.). Consulté le Mai 12, 2009, sur .Net Scraps | ASP.NET, Sharepoint, IIS, Troubleshooting, Tips and Tricks !: http://www.dotnetscraps.com/dotnetscraps/post/Explaining-Delegates-in-C-Part-3-(Events-again).aspx
- Le guichet du savoir. (s.d.). *L'oeil humain*. Consulté le Mai 12, 2009, sur Le guichet du savoir: http://www.guichetdusavoir.org/ipb/index.php?showtopic=12506
- MorpionMx. (s.d.). *[C#] OPERATIONS CROSS THREADS*. Consulté le Mai 12, 2009, sur CSharpFR.com: http://www.csharpfr.com/tutorial.aspx?ID=174
- *New European Driving Cycle - Wikipedia, the free encyclopedia*. (s.d.). Consulté le Mai 12, 2009, sur Wikipedia: http://en.wikipedia.org/wiki/New_European_Driving_Cycle
- Nico-pyright. (2007, Décembre 5). *Tutoriel : Travailler avec les fichiers de configuration en C#*. Consulté le Mai 5, 2009, sur developpez.com: http://nico-pyright.developpez.com/tutoriel/vc2005/configurationsectioncsharp/
- nowhereelse.fr. (s.d.). *http://www.nowhereelse.fr/wp-content/uploads/2009/02/.* Consulté le Mai 12, 2009, sur http://www.nowhereelse.fr/: http://www.nowhereelse.fr/wp-content/uploads/2009/02/
- OPC FOUNDATION. (s.d.). *Using OPC via DCOM.* Consulté le Mai 12, 2009, sur http://www.opcfoundation.org/Archive/c9cf0fff-65fa-4362-bc93-aa7d0ab58016/Using%20OPC%20via%20DCOM%20with%20XP%20SP2%20v1.10.pdf
- OpenWinforms. (s.d.). *Digital display control in C#.Net using GDI+*. Consulté le Mai 12, 2009, sur http://www.openwinforms.com/digital_display_control_net.html
- OpenXML developer. (s.d.). *OpenXML developer : Creating spreadsheets from a server appplication*. Consulté le Mai 12, 2009, sur OpenXML developer: http://openxmldeveloper.org/articles/Creating_Spreadsheets_Server.aspx
- Rotronics. (s.d.). *autoscan*. Consulté le Mai 12, 2009, sur Rotronics: http://www.rotronics.com/fr/html/autoscan.html
- Rotronics. *Manuel d'utilisateur de la station météo.*

- Wikipedia. (s.d.). *Framework .NET*. Consulté le Mai 12, 2009, sur Wikipedia: http://fr.wikipedia.org/wiki/Framework_.NET
- ZedGraph. (s.d.). *Speed up the redraw time - ZedGraphWiki*. Consulté le mai 12, 2009, sur ZedGraphWiki: http://zedgraph.org/wiki/index.php?title=Speed_up_the_redraw_time

12 Liste des figures


Province
de Liège


campus
automobile
SPA-FRANCORCHAMPS
centre de compétence

13 Notes


Province
de Liège


campus
automobile
SPA-FRANCORCHAMPS
centre de compétence

14 ANNEXE